NordDesign 2004 – Product Design in Changing Environment
18-20 August 2004, Tampere, Finland

COMPARATIVE ANALYSIS OF PRODUCT MODULARISATION METHODS

Katja M. M. Hölttä

Canter for Innovation in Product Development
Massachusetts Institute of Technology
30 Memorial Dr. RM E60-246
Cambridge, MA, 02139
USA
Visiting from: Machine Design, Helsinki
University of Technology
E-mail: katja.holtta@hut.fi

Keywords: modularisation, product architecture, product structuring

Abstract

Modularity has many benefits. Several methods have arisen to create modular products. In this paper three modularity methods that have been well established in academia and used in industry are compared against one another. These are Function Structure Heuristics, Design Structure Matrix, and Modular Function Deployment. These methods are applied to a cordless drill to show how they perform for a single product. In addition, they are applied to a saw of the same family of products to see how the methods work for a product family. We find that the methods all give different suggestions for a modular architecture for the drill. We also find that the methods give suboptimal results in the case of a product family. We also tested the repeatability of the methods. It varied between 60% and 90%.

1 Introduction

The module definition used in this article is adapted from various sources [Baldwin&Clark00, Ericsson&Erixon99, Ulrich&Eppinger04]. A module is a structurally independent building block of a larger system with well-defined interfaces. A module has fairly loose connections to the rest of the system allowing an independent development of the module as long as the interconnections at the interfaces are carefully considered.

Recent literature has shown many advantages of modularity. It provides advantages in terms of scale and scope, economies in parts sourcing, and support for mass customisation [Baldwin&Clark00]. Modularity makes product architecture easier to manage [Ericsson&Erixon99]. Modularity also eases end-of-life processes such as recycling of parts [Newcomb et al.98]. Modularity provides flexibility that allows both multiple product variations and technology development by the changing of modules, without requiring changes in the overall product [Ericsson&Erixon99, Ulrich&Eppinger04].

Many methods have been developed to create modular products and to benefit from the advantages of modularity, but it is unclear what method to use and when. Since all methods have the same goal, one could presume that the methods give same or similar suggestions for a modular architecture. Often, a modularity method is introduced through an example of modularising a single product, but since one of the main purposes of modularity is to use identical modules in multiple products in a product family, the method should work also for product families and not just single products. The purpose of this paper is to test whether different methods results in a common modularisation, whether the methods work for product families, and how objective or subjective the methods are.

In this paper, three modularity methods: Function Structure Heuristics [Stone et al.00], clustering of a Design Structure Matrix [Pimmler&Eppinger94], and Modular Function Deployment [Ericsson&Erixon99] are compared by applying them all to the same products and comparing the results. Strengths and weaknesses of each method are identified. Commonalities between methods are discussed. The analysis will also show what types of functions are handled in a similar way by all methods. The methods are applied both to single products and product families and the repeatability of the methods is also discussed.

2 Modularity methods

Three modularity methods were chosen for the comparison. These three methods are well established in academia and used in industry. There are also other methods but they are excluded from this study since they have either been introduced only in a single publication or have not been used in industry.

2.1 Function Structure Heuristics

Function Structure Heuristic method was developed by Stone and his colleagues [Stone et al.00]. It is based on Pahl and Beitz's function structures [Pahl&Beitz99]. A function structure is a functional decomposition block diagram of all the product's functions and the material, energy, and information flows between them. Stone et al. identify modules from a single product's function structure by finding the *dominant flow*, *branching flows*, and *conversion-transmission* function pairs [Stone et al.00] using three corresponding heuristics. Zamirowski and Otto [Zamirowski&Otto99] present three additional heuristics to find common modules across products in a product family. They identify *similar and repetitive* functions within a single product, *common functions* across products, and *unique functions* that are found only in one product and separate them into modules. A good tutorial of the method is given by [Otto&Wood01].

The idea of the Function Structure Heuristic method is to consider the many possible alternative modules that can be defined according to the heuristics. While the heuristics define possible modules, it is up to the designer to choose which ones make sense. Further, the heuristics are maximal heuristics. They state only that one should not define modules larger than indicated. For example, any module defined as a serial chain of functions by a dominant flow, can be subdivided in any way and still be consistent with the heuristics. As such, the approach provides modularity suggestions only; it is not a unique algorithm. Therefore, designer insight and good judgement can enter the process. This is either a benefit or a problem, depending upon one's perspective.

Three heuristics apply to single products and three to product families of similar products. The main modularisation factors considered by the Function Structure Heuristics are module interfaces and functionality. Other factors such as business or strategy related criteria are not represented in the Function Structure Heuristic method, but are left up to designer judgement.

2.2 Design Structure Matrix

The most common application of the DSM [Ulrich&Eppinger04] is to organise product development tasks or teams to minimise unnecessary rework and thus help manage and speed up the development process. The DSM can also be used to define modules within a single product's architecture. In the component, or function, based DSM, also called architecture DSM, components or functions are placed on the row and column headers of the matrix. Components or functions are then mapped against each other and their interactions are marked in the matrix [Pimmler&Eppinger94, Blackenfelt01].

Once functions or components and their interactions are placed in the DSM, a clustering algorithm is applied to group the functions or components so that the interactions within clusters are maximised and between the clusters minimised. The formed clusters are possible module candidates. There are many algorithms and one can develop one's own to suit the needs of a specific case. The basic idea of a clustering algorithm is to reorder the rows and columns so that all marks are as close to the diagonal as possible or form a tight cluster with other marks. The algorithm used here is developed by [Thebeau01]. This was chosen because it is a well-defined computerised algorithm. The algorithm can result in overlapping modules or it may leave one or more functions out of the final clustering, in which case it is up to the designer to decide how to deal with them. An overlapping section could be duplicated and placed in both modules or forced to be only in one of the modules where the algorithm suggested it could be. For more about the component based DSM method, refer to [Browning01].

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

2.3 Modular Function Deployment

Another modularisation method, more management- and less engineering-oriented, is MFD [Ericsson&Erixon99]. It is also based on functional decomposition, but in this method, modularity drivers other than functionality are considered. MFD is designed to modularise a single product. There are twelve modularity drivers in MFD. The first is *carryover* i.e. a specific function will carry over to different products and no technology changes are expected. The next two, *technology evolution* and *planned product changes*, take both unexpected and expected changes into account. *Different specification* enables product variation and *styling* considers how the modularity choice would affect the appearance of the product. *Common unit* is similar to Zamirowski's common function heuristic in the Function Structure Heuristic method. *Process and/or organisation*, *separate testing*, *supplier availability*, and *service and maintenance* are related to the organisational effects of modularisation. *Upgrading* allows future additions to the product. *Recycling*, the last modularity driver, considers the afterlife of the product. To apply this method one or a few of these modularity drivers are chosen according to the firm's strategy. Ericsson and Erixon offer a good tutorial on the method [Ericsson&Erixon99].

MFD is similar to Quality Function Deployment (QFD) [Hauser&Clausing88] but here modularity drivers are mapped against functions in a module driver profile instead of customer requirements in a matrix. The grouping into modules is started with the functions receiving the highest summed scores; and the functions dominated by the same modularity drivers are good candidates for a module according to this method.

MFD suggests that the ideal number of modules is approximately the square root of the number of parts or assembly operations. The estimate is based on optimising the assembly lead-time of the whole product. [Ericsson&Erixon99]

3 Approach

Twenty engineers and graduate students in engineering with either a Bachelor's or Master's degree in Mechanical Engineering at Helsinki University of Technology were used to apply the different modularity methods. In addition, we tested the methods on four other, also electro-mechanical, products using a separate group of 20 engineers and students [Holtta&Salonen03], which brings the total number of engineers and students to 40 and the number of products to 6, of which 4 were two pairs of products in a product family. The modularisation schemes are shown here for a cordless drill and a saw.

Engineers and students were familiarised with the products and the methods beforehand. They were also allowed to keep a reading package about the modularity methods during the experiment. Each engineer and student completed the modularisation of the products independently, without any external influence on the modularisation choices. If questions arose, the subjects were guided only by instructions relating to the methods, not on how to make the modularisation choices. This was done to ensure that the given instructions did not have an effect on the results of this research.

[Kurfman et al.00] have shown function structures to comprise a good representation of product architecture, therefore the same function structure decomposition (Figure 1) was used for all three methods to have a common starting point for all methods. Similar function structure was used for the saw [Otto&Wood01].

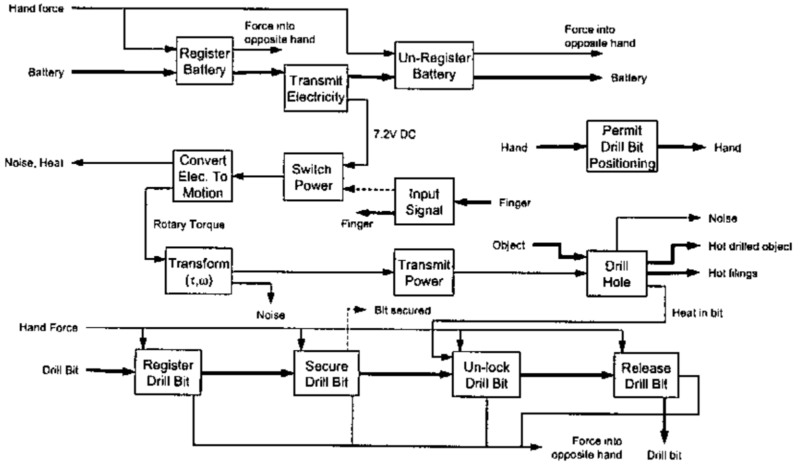

Figure 1 Cordless drill function structure [Otto&Wood01]

The function structure could be used as such for the Function Structure Heuristic method. The function structure was handed to the engineers and students and they applied the Function Structure Heuristics on them. The dominant flow, branching flow, and conversion transmission pair heuristics were applied for the single product example and the common, unique, and similar function heuristics were applied to the product family example. The function structure was converted into a matrix for the DSM method. This was done by listing the product functions as headers in the rows and columns of a matrix. Interactions between functions were marked in the corresponding intersecting cells of the matrix. A clustering algorithm developed by [Thebeau01] and found at the DSM website[1] was used to cluster the functions into modules. For the MFD, the function structure was converted into a *module indication matrix*, where the functions are the column headers of the matrix. The 12 modularity drivers are the row headers. The module indication matrix was filled to show the connections between the functions and the modularity drivers. We did not have access to the company itself, so the matrix was filled based on a careful examination of the products. Engineers and students defined product modules in the module indication matrix.

The different modularisations of all the participants were gathered and the most common solution was formed weighing the more experienced engineers' answers over the students. These common solutions are presented in the results section. Then, each engineer's or student's matrix was compared to the common solution by calculating the percentage of functions modularised differently. For an overall repeatability the percentages were averaged and subtracted from 100% resulting in a repeatability percentage of each method.

4 Results

Single products

As predicted by the results of the previous study [Holtta&Salonen03], the three methods all give a different suggestion for possible product modularisation of the cordless drills (Table 1). All methods identify certain groups of functions, that should be combined into a module, in some particular way, but they do not agree on how many other functions these so-called *module cores* should have. One common observation in this study, as well as in looking at the results of the previous study, is that all methods identify the product's drive unit as a module. The drive unit is typically a central part of a product and all methods suggest it should be bundled up as a module. However, the methods do not agree on the size of the module, i.e. what functions should be included in the drive unit module.

[1] www.dsmweb.org

Table 1 Cordless drill modules by different modularity methods (same colour and pattern define a module. * denotes a module is split in the table)

MFD	Conversion transmission	Branching	Dominant	DSM
register battery	register battery	register battery	register battery	register battery
unregister battery	unregister battery	unregister battery	unregister battery	unregister battery
transmit electricity	transmit electricity	transmit electricity	transmit electricity	transmit electricity
switch power	switch power	switch power	switch power	switch power
conv electr to moti	conv electr to moti	conv electr to moti	conv electr to moti	conv el to motion
transform	transform	transform	transform	transform
transmit power	transmit power	transmit power	transmit power	transmit power
drill hole	drill hole	drill hole	drill hole	drill hole
input signal	input signal	input signal	input signal	input signal
register drill bit	register drill bit	register drill bit	register drill bit	register drill bit
secure drill bit	secure drill bit	secure drill bit	secure drill bit	secure drill bit
un-lock drill bit	un-lock drill bit	un-lock drill bit	un-lock drill bit	un-lock drill bit
release drill bit	release drill bit	release drill bit	release drill bit	release drill bit
permit drill posit	permit drill posit	permit drill posit	permit drill posit	permit drill posit

The three methods identify no common modules in the cordless drill. The register/unregister battery module is identified by both the DSM and the MFD, input signal module by MFD and branching flow, and lastly the permit drill positioning module is identified by both the DSM and the dominant flow. All other modules are different. All methods find similar modules, but the final solutions are very different. The most common module identified by the methods is the chuck module i.e. functions "register drill bit", "secure drill bit", "un-lock drill bit", and "release drill bit". It is identified as one module by the Function Structure Heuristics. MFD identifies a similar grouping into a module, but the function "permit drill positioning" is added since it has a similar module driver profile as the chuck functions. The DSM, on the other hand, splits the chuck into two separate modules and leaves the "permit drill positioning" out as a single function module.

It is not clear what method gives the "best" modular architecture, since what is best depends also on other factors such as degree of desired modularity, supplier choices, performance, etc. However, some conclusions can be drawn. MFD is a fairly systematic and DSM an automated modularisation process. This makes it fairly easy to interpret the results compared to the overlapping suggestions given by the three single-product heuristics of the Function Structure Heuristics. Although, one should notice that the DSM can sometimes give overlapping modules as well, as shown in [Holtta&Salonen03]. However, since human judgement must be used while applying the heuristics, they suggest reasonable modules. MFD, on the other hand, puts the "permit drill positioning" function (which is basically the drill handle and casing) into the chuck module. The DSM also suggests counterintuitive results such as splitting the chuck module into two parts; one for registering and securing the drill bit and another for un-locking and releasing the drill bit.

Furthermore, each method suggests a different number of modules. MFD has a rule to limit the number of modules and the Function Structure Heuristics are designed to be maximal heuristics and therefore they give a lower number of modules than the DSM.

Product family

When the three methods were applied to a family consisting of a drill and related saw, we found that the Function Structure Heuristics were better suited to a family than the MFD and the DSM (Table 2). The two products in the example are very similar, so the MFD performed for a family better than in our earlier work [Holtta&Salonen03]. In a typical product family, most functions are not the same, as they are in this example. For example in a kitchen appliance family, only some user interface buttons and general colour and shape might be the same across the entire family. In addition, a few modules could be shared in several products, such as a hand blender handle that can be turned into several products by attaching e.g. a chopper or blender module to it. The drill and saw case is therefore atypical since we only picked two very similar products in the family and therefore some good common module candidates were identified as common modules for the product family.

One barrier to common module identification in the MFD could be another driver conflicting with the *common unit* driver in the MFD. Also the DSM algorithm grouped a few possible common module candidates in each product differently. The algorithm optimises one product at a time, which can lead to suboptimal modular architecture for a product family. As expected, the Function Structure Heuristics method's family heuristics [Zamirowski&Otto99] performed the best. It is the only method specifically designed for a product family. We only show the *common function* heuristics' results in Table 2. In addition, the *unique function* heuristic identified the remaining functions not modularised by the *common function* heuristic as unique modules and the *similar or repetitive* heuristic found no similar or repetitive functions within each product.

One reason why the methods worked better in this study than in the previous study, is that the products in this study are based on a platform and the products in the first study were not. It is interesting that even though one of modularity's main ideas is to create economies of scale and provide variety with a set of common and unique components, many modularity methods have been developed mainly for single products. This suggests that other or modified methods should be used for product family development. If a product platform exists already, the methods perform better.

Table 2 Modules for the drill/saw –family (same colour and pattern define a module. * denotes a module is split in the table)

MFD drill	MFD saw	DSM drill	DSM saw	Common functions drill	Common functions saw
register battery	register battery	register battery	register battery*	register battery	register battery
un-register battery	un-register battery	un-register battery	un-register battery*	un-register battery	un-register battery
[illegible]	[illegible]	[illegible]	transmit elect*	transmit elect	transmit elect
[illegible]	[illegible]	[illegible]	switch power*	switch power	switch power
[illegible]	[illegible]	conv el to motion	[illegible]	conv ele to motion	conv ele to motion
[illegible]	[illegible]	transform	[illegible]	transform	transform
	[illegible]		[illegible]		transform motion
[illegible]	[illegible]	transmit power	transmit power	transmit power	transmit power
drill hole	cut board	drill hole	cut board	drill hole	cut board
input signal	input signal	[illegible]	input signal*	input signal	input signal
register drill bit	register blade	register drill bit	register blade	register drill bit	register blade
secure drill bit	secure blade	secure drill bit	secure blade	secure drill bit	secure blade
un-lock drill bit	un-lock blade	un-lock drill bit	un-lock blade	un-lock drill bit	un-lock blade
release drill bit	release blade	release drill bit	release blade	release drill bit	release blade
permit drill posit	permit blade posit	permit drill posit	permit blade posit	permit drill posit	permit blade posit

Repeatability

The repeatability of the methods varied between 60% and 90%, *conversion-transmission* heuristic being the strongest and the *dominant-flow* heuristic being the weakest (Table 3). The repeatability correlates to the objectivity or subjectivity of the method. The more repeatable a method is, the more objective it is. The repeatability of the DSM is not included since it is a computer run algorithm. We ran the algorithm multiple times, and if started with the same order of rows and columns for each run, the results were 100% repeatable. The modularisation scheme suggested by the DSM algorithm depends on the order of the rows and columns.

Table 3 Repeatability of the modularity methods

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

The results here are in accordance with the previous study [Holtta&Salonen03] for the most part. The only major differences were the repeatability of the MFD and the *dominant flow* heuristic. For MFD the previous study showed a repeatability of 68% but in this one it was 85%. The difference is partially due to the fact that students were better familiarised with the product and the method than before. The familiarisation lead to choosing a modularisation that was very similar to the existing modularisation. Only the number of modules was reduced. Four original modules (the drill bit, the chuck, the switch, and the battery) remained unchanged. The only difference was that the rest of the functions were combined into a single module instead of multiple modules. This suggests that either the MFD drives toward the old solution or the current modularisation was already good. The low repeatability of the *dominant flow* heuristic, on the other hand, is due to the vague definition of the heuristic. Many students and engineers complained about the difficulty of interpreting the heuristic, which was reflected in the low repeatability of the method.

The repeatability can be applied in practice to permit use of the modularity methods. That is, an engineer can easily follow the modularisation rules until about 70% of the product is modularised. At that point, use of any of the methods diverge and so engineering judgement comes to play. At that point, the methods do not give definitive answers.

5 Conclusions

In this paper three modularity methods were analysed by testing them on a cordless drill and a saw. We found that all methods give different suggestions for a modular architecture of the product. Further, the number of modules suggested by each method was different. Clearly, there is room for improvement in the modularity methods. The methods group the major functions into *module cores* but provide little insight what to do with the rest of the functions to fully modularise a product. The goal of each method is to provide the benefits of modularity mentioned earlier, but further research is needed to decide which method best fulfils the expectations or if a new method should be developed.

Another interesting factor is that even though modularity is usually discussed in the context of product families, many modularity methods have been developed for optimising a single product and are therefore not well suited to designing a product family.

We also analysed the repeatability of the methods. It varied between 60% and 90%. We concluded from this that each method helps a designer to modularise about an average of 70% of the functions in a given product, and in order to come up with the final modularisation scheme, engineering judgement must be used.

Yet another factor in modular design is the feasibility of the modularisation schemes suggested by the methods. A more complete and more objective method (such as the DSM) tends to lead to some infeasible modules, which is not a problem with a more engineering-judgement-based method (such as the Function Structure Heuristics). But the more subjective a method is, the more laborious and less repeatable it becomes when used with complex problems.

References

Baldwin, C. Y. and Clark, K. B., "Design Rules: The Power of Modularity", MIT Press, Cambridge, MA, 2000.
Design Structure Website. http://www.dsmweb.org/. (viewed 8.12.2003)
Blackenfelt, M., "Managing complexity by product modularisation", *Doctoral Thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm, 2001.*
Browning, T. R., "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," IEEE Transactions on Engineering Management, 48, 3, 2001, 292-306.
Ericsson, A. and Erixon, G., "Controlling design variants: Modular product platforms", ASME Press. New York, NY, 1999.
Hauser, J.R., and Clausing, D., "The House of Quality", Harvard Business Review, 1988, 63-73.
Holtta, K and Salonen, M. "Comparing three modularity methods". ASME design engineering technical conferences DETC 2003. Chicago, IL.
Newcomb, P. J., Bras, B., and Rosen, D. W., "Implications of modularity on product design for the life cycle". Journal of Mechanical Design, 120, 1998, 483-490.
Otto, K. and Wood, K., "Product design: techniques in reverse engineering and new product development", Prentice Hall, Upper Saddle River, NJ, 2001.
Pahl G. and Beitz W.,"Engineering Design", Springer-Verlag, London Ltd, 2nd ed, 1999.
Pimmler, T. U. and Eppinger, S. D.,"Integration Analysis of Product Decompositions", ASME design engineering technical conferences DETC 1994, Minneapolis, MN, 1994.
Stone, R. B., Wood, K. L. and Crawford, R. H., "A heuristic method for identifying modules for product architectures", Design Studies, 21, 1, 2000, 5-31.
Kurfman, M., Stone, R., Van Wie, M., Wood, K., Otto, K.,"Theoretical Underpinnings of Functional Modeling: Preliminary Experimental Studies", ASME design engineering technical conferences DETC 2000, Baltimore, MD.
Thebeau, R. E., "Knowledge Management of System Interfaces and Interactions for Product Development Processes", MIT Masters Thesis, System Design & Management Program, 2001.
Ulrich, K. T. and Eppinger, S. D., Product Design and Development, McGraw-Hill, New York, NY, 3nd ed., 2004.
Zamirowski, E. J. & Otto K. N. "Identifying Product Family Architecture Modularity Using Function and Variety Heuristics", ASME design engineering technical conferences DETC 1999, Las Vegas, NV, 1999.