Publication V

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

DOI:10.1016/j.destud.2004.10.001

Incorporating design effort complexity measures in product architectural design and assessment

Katja M. M. Hölttä, MIT Center for Innovation in Product Development, 30 Memorial Dr. RM E60-246, Cambridge, MA 02472, USA. Visiting from: Helsinki University of Technology, Department of Machine Design
Kevin N. Otto, Product Genesis, Inc. 245 Bent St, Cambridge, MA 02141, USA

A flexible product architecture enables easy future changes to the product. We introduce here a procedure to design products using existing modularity methods and a novel redesign effort complexity metric that helps define module boundaries so that changes in the modules require minimum redesign effort. The metric is based on redesign difficulty of material, energy, and information flows. We find that different interface types need different amounts of redesign effort in order to accommodate a change. We show the use of our method through an example of a gas sensor.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: conceptual design, design methodology, engineering design, interface design, modularity

There are always changes to a product. A product may need design iterations during the original development or it may need change due to customer complaints. In addition, a new version of a product or another product for the same family is often developed based on the original product. Whatever the situation, a maker of a flexible product that is quick to change is more likely to succeed than a competitor who is unprepared for changes.

Modularity is, due to its clear interfaces, one common way of providing flexibility that enables product variations and technology development without changes to the overall design (Ericsson and Erixon, 1999). One important factor of a good module is the ease with which a module design can be redesigned without impacting its interfaces and the rest of the product. In this paper we develop a method to define module boundaries and relatively compare proposed product architectures on

Corresponding author:
K. M. M. Hölttä
holtta@mit.edu

design effort complexity, as measured by design difficulty to change the interfacing flows between modules.

The remainder of the paper is structured as follows. We will describe our approach as well as the research method used in the following two sections. The next section will present the results. We will then show through an example how to use our method. We will end the paper with conclusions.

1 Background

1.1 Modularity

A *module*, as it is used in this paper, 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.

One feature of good product modularity is the ease with which modules can be changed within a product i.e. their degree of isolation. Yet, there are few methods to quantify modularity and to thereby choose module boundaries. Stone et al. (2000) developed a heuristic method to identify modules by finding the dominant flow, branching flows, or conversion—transmission function pairs within a function structure. Otto (2001) uses the heuristics to create a modular product family based on cost and revenue analysis of the architecture over customer requirements. Ericsson and Erixon (1999), on the other hand, developed a method called modular function deployment that groups functions according to such strategic aspects as technology evolution, planned changes, or styling. Blackenfelt (2000) combines Ericsson and Erixon's method with the design structure matrix (DSM) (Ulrich and Eppinger, 2004) to cluster technical solutions into modules based on their estimated interaction strengths. Also, Pimmler and Eppinger (1994) use the DSM to cluster functions into modules minimizing the connections between modules.

Holtta and Salonen (2003) showed, interestingly, that when the methods introduced above are applied to solve the same modularization problem, they all give different results. This is due to the different criteria used in each method. The methods help the designer to modularize approximately 70% of the functions and the rest is left up to the design engineer. Engineers need decision support with such preliminary design activity, particularly here in grouping functionality into modules (Stake and Blackenfelt, 1998; Smith et al., 2001). Current modularization methods help identify module'cores' i.e. functions around which a module is built. A module'core' consists of the main functions a method has clearly assigned for a module. In addition, there are functions that could be placed in more than one module without breaking the method's rules. Ericsson and Erixon (1999), in their modular function deployment, recognize the importance of a step for interface design, but the discussion does not present a specific tool. We find that without proper decision support, engineers look at the interfaces from a distance through the lens of their specific experience, which may not be sufficiently general, disregarding the interaction types as instantiated by various flows though different possible interface boundaries. This might lead to unnecessarily complex module boundaries that are inflexible for future changes. In summary, existing methods use subjective assessments or qualitative attributes to identify modules and to evaluate different architectures. Many decisions depend on the designer. This affects the repeatability of the methods.

1.2 *Interface complexity*

Architecture assessment must include many factors, such as ease of upgrades, ease of supporting variants, design ease, supplier capability (Mikkola, 2000; Camuffo, 2001), and manufacturing support, for example (Ericsson and Erixon, 1999). Among these, one important factor is design effort complexity, the ease with which a sub-system can be redesigned without impacting its interfaces and the rest of the product. In hardware design, the standard interfaces ease independent development of modules, but the information passed through the standard interface may still require changes in the adjacent module. In good software design, modules are often defined so that they 'hide' (Parnas, 1972; Bass et al., 2003) information to prevent changes from affecting other modules. Similar tactics should apply for mechanical and other design as well since redesign is often unavoidable. A new product variant or an upgrade should require as little redesign effort as possible. Therefore a redesign effort based criteria is important in the architecture assessment process.

Research in design complexity primarily considers design process modeling and design process complexity and not so much redesign effort and interface complexity. For example, Suh (2001) defines complexity as the probability of achieving the requirements. His work also requires design process modeling to be able to define his complexity measure in a meaningful way. Further, he does not specifically address the problem of defining modules. Braha and Maimon (1998), on the other hand, discuss artifact complexity, where they argue that the best artifact design in terms of complexity is one that has the minimum information content and is most likely to meet its required specifications. El-Haik and Yang (1999) discuss additional mathematical representations of Suh's axioms and calculate complexity of an engineering design. All these design complexity measures discuss the overall complexity of a design. They are not suitable for our purposes for two reasons: (1) We work to define module boundaries, the ideal interfaces (in terms of design effort) between modules; the main goal is not a single number to represent all criteria upon which to evaluate the complexity of a design, but rather we focus on the multiple facets of design effort complexity. (2) These methods evaluate designs at the later stages of the design process where the design process structure is understood, whereas we aim to ease the fuzzier front end before any project planning has been undertaken. We seek a tool to draw module boundaries and evaluate architecture concepts using a representation of the design only, and with minimal estimates of the design process activity.

There is some literature on estimating interface complexity in the early phases of product development. Braha and Maimon's (1998) structural complexity measure consists of the number of relations and functions in an artifact, as well as the number of distinct relation and function types in an artifact. Blackenfelt (2000) describes a similar complexity definition in the context of modularity. He however does not focus upon design difficulty. Also Maier and Rechtin (2000) describe architecture complexity by the amount of connections, or communication in case of software, between modules. Also an IEEE standard (IEEE Std 982.1-1988) exists for software that defines complexity based on the inputs and outputs of a module. These are similar in that they all treat any relation as having the same difficulty, which is not generally the case. There are no means in existing literature to compare component interaction types to properly evaluate module boundaries. Looking at the number of interactions at each interface is not enough. For example, which is more difficult to compensate, a 20% increase in information bandwidth or a 20% increase in operating voltage? As we will demonstrate by developing a redesign effort complexity metric, some interaction types are more difficult to modify than others.

We would like to point out that our redesign effort metric is not meant for deciding the number or size of the modules alone. Our metric along with others, such as assemblability, cost, suppliers, team size (Braha, 2002), etc., are all important criteria to use in such a multi-criterion decision. In addition, emergent properties such as cost, weight and performance type criteria must be considered (Ulrich, 1995; Gupta and Krishnan, 1999; Gonzalez-Zugasti et al., 2001; Krishnan and Gupta, 2001; Whitney, 2004). We provide a structured means to represent one metric – design effort complexity – in such a decision. The final decision can be made e.g., using optimization (Gonzalez-Zugasti et al., 2001), Pugh chart concept selection (Otto, 2001), etc. approaches. There are other approaches to estimate the number of modules one should use in a design. For example, Ericsson and Erixon (1999) develop the ideal number of modules for mechanical assembly purposes as approximately the square root of the number of parts to be assembled. Similarly Braha (2002) suggests using the connection of product development teams and tasks to product modules to determine product modules based upon minimizing organizational difficulty. He partitions tasks to teams and limits their sizes by minimizing the time needed for communication between teams and considering how many design attributes a team can handle. The same partitioning could drive the module size as well in order to keep the architecture aligned with the organization. These and others are all valid concerns that should, as well as our metric, be included in the overall decision on the number and size of modules.

2 *Approach*

We introduce here a six-step procedure to modularize a product and to identify and evaluate module boundaries (Figure 1). This procedure will help in designing products that are flexible to change. The core of the method is in the steps 4 and 5, where we construct a measure of the design effort complexity of the module boundaries in terms of design effort to change the boundaries i.e. module interfaces.

The method starts by customer need identification. The needs can then be used either to build a function structure or to evaluate a function structure built based on an existing product. Kurfman et al. (2003) show that function structures result in quasi-unique product representations and that the functional basis vocabulary (Hirtz et al., 2002) improves the functional modeling making it roughly 80% repeatable. Kurfman et al. (2003) also show that method works for both redesign and an original product, but benefits are clearer with redesign projects. We use a similar approach in our method to overcome the subjectivity of function structures.

To build a function structure for a new product, one can follow the procedure developed by Otto and Wood (2001) for steps 1 and 2. They start similar to our procedure by identifying customer needs and

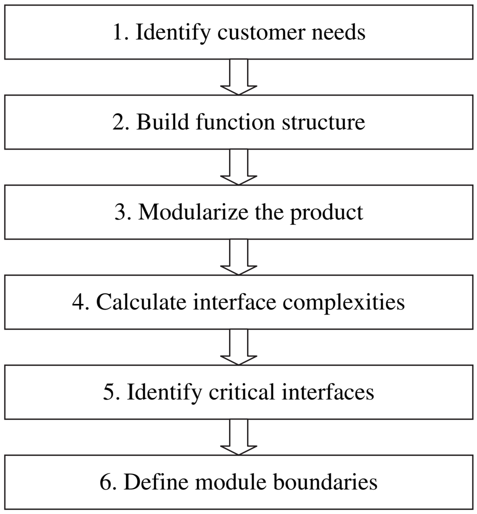

1. Identify customer needs
2. Build function structure
3. Modularize the product
4. Calculate interface complexities
5. Identify critical interfaces
6. Define module boundaries

Figure 1 The procedure

transforming them into function chains that are weighted with the importance to the customer. This is especially useful for new product concepts. In case of an existing product, however, we find that one can start by decomposing the product into functions, each function representing a sub-system of the product, and building up a function structure according to the assembly decomposition. When completed, one should identify the function chains that satisfy the most important customer needs to ensure that all customer-critical flows are well identified.

The next step is to modularize the product. Here, the product is modularized according to the criteria that is important to the company e.g., life cycle issues or project management. In this third step, modularize the product, one can choose from many methods introduced before. One can use the product decomposition of the function structure regardless of the modularization method. We choose here to use a binary DSM clustering algorithm (Thebeau, 2001) as an example. As we argued before, a modularization method identifies only module' cores' and therefore additional steps are needed.

For the fourth step, calculate interface complexities, we have developed a novel approach to evaluate the design effort complexity of an interface. We use the function structure flows to represent interactions between functions. Fixson (2001) suggests that interactions have different intensities. He also points out that different connections have different degrees of reversibility and this should affect the complexity of the interactions of the module to the rest of the system. We quantify this by deriving a metric that represents the difficulty that different module boundaries would have in terms of redesign effort in changing the values of the flows, measured using estimates of required redesign hours. The interface complexity metric can be calculated for various flow types, as will be shown in the following sections.

The calculations can be used to identify critical interfaces in a product architecture for the fifth step. The larger the design effort complexity metric on a specific interface, the better it is to keep the interface within a module. And similarly, the smaller the design effort complexity metric at an interface, the better candidate the interface is to be at a module boundary. These are analogous to the tactics in software to aim to keep the high-bandwidth communication within a module and place low-bandwidth links between modules (Bass et al., 2003). This eases the development of the modules since a team developing a module is more likely to handle the complex interfaces than if the interface was to be designed by two separate teams developing separate modules. This is also supported by Sosa et al. (2000).

The sixth and final step is to define the module final boundaries. Our design effort complexity metric is one measure to be factored in with the many other metrics one must consider when establishing boundaries. In summary, we first built the function structure in steps 1—2 and then in step 3, we defined the strategic, organizational, etc. modules. Steps 4 and 5 analyzed the module interfaces and the last step was to redefine the module boundaries according to the design effort complexity metric without conflicting with the original strategic (or other) modularization. Our design effort complexity metric can help choose the best alternative from various modularization schemes in terms of minimizing the design effort at an interface.

3 *Methodology*

We conducted multiple interviews in two case studies in order to evaluate the design effort complexity of different interface types. On these case studies, we examined dozens of modules and asked experienced practicing design engineers how long it would take them to both redesign this module at larger or smaller capacity and incorporate that as a redesign into the original product as compared to the original effort. We did this with a total of 11 engineers from three different companies involved in designing the four products we chose to study here. Case study 1 consisted of two companies designing two products together. We chose 14 modules from these products and received answers from the five engineers interviewed to 61 questions related to the changing of several inputs and outputs. Case study 2 consisted of one company and two of its products. We chose 10 modules and received answers from six engineers for 47 questions related to the changing of inputs and outputs.

Many factors can impact design time, notably experience level, familiarity, and overhead of any particular corporate culture. Therefore the posed questions themselves were not so straightforward to extract useful data. We accounted for these sources of error in two ways. First, we only worked with highly experienced system engineers who did design work on different design variants of the products in question. All had many design experiences with each of the module types we posed questions on. Second, we selected engineers with all mechanical, electrical, and software backgrounds to eliminate the biases toward own paradigms. We also conducted the cases fully independent of one another to see if there are company or product type specific as well as universal effects of redesign.

The product domain we explored was the general domain of electro-mechanical devices. We chose this since they are highly engineered, amenable to a function structure systems representation, and design experts were available. For the first case study, we chose within the medical device domain two slightly different injector heads used to inject contrast into a patient's bloodstream during imaging procedures. These were real FDA approved products on the market bearing revenue. The injectors were jointly developed by two companies working collaboratively. For the second case study, we chose two process sensor systems, also on the market bearing revenue, developed by yet a third, completely unrelated company. This company had no contact or associations with the previous two. For all, we first built function structures for the product architectures. We decomposed the products to the assembly level of the manufacturer. We also assigned each component or sub-system a function e.g., a motor was labeled 'convert electricity into rotation'. We then represented all the connections between the components or sub-systems with their material, energy, and information flows. For example the motor torque going to the transmission was represented with an (mechanical) energy flow of torque/rotation. As discussed elsewhere (Otto and Wood, 2001), one should not forget supporting functions and flows such as vibration and damping or support of weight when completing the function structure. The law of energy conservation should apply in a complete function structure.

Once the function structures were complete, we took each of the functions/sub-systems represented by each block in the diagram, and analyzed the relative redesign effort required to change each flow entering and exiting the sub-system. We performed the same procedure separately for both case studies.

The two injectors in the first case study consist of approximately 30 functions and 40 inter-function connections. The two sensor systems in the second case study were slightly more complex, one having 37 functions and 76 inter-function connections and the other 32 functions and 58 inter-function connections. One could use our approach at different abstraction levels of the architecture but our scope is the assembly level of a product manufacturer. Similarly we leave the hierarchical aspects of flows at different abstraction levels outside this study.

The flows into- and out-of-a-function (sub-system) make for an effective representation of the interaction and primary interface physics. The flows were classified according to functional basis similar to Hirtz et al. (2002) (see Table 1). We used energy flows of electrical, mechanical, thermal, and pneumatic, information, and material flows. One should note that this classification conveys the primary intention of the flow. That is, an information flow can be both an electrical flow and an information flow and vice versa. All material flows contain energy. We chose to represent power connections as electrical flows and connections with control information as information flows. If an information flow is a simple on/off electrical connection, we represent it with only an electrical flow. Material flows also represent the energy they contain, until an extraction function extracts it, for example. As an example, one could look at the electrical energy flow from function *convert* ϵ_r to C to *transmit C value* (Figure 4). This flow carries the capacitance information but since it is such a simple electrical flow, it is interpreted as an electrical flow.

Our hypothesis is that this classification serves as an indicator of interface difficulty. Each flow type is parameterized with a small number of descriptors such as voltage, speed, force, etc. This is similar to Boothroyd et al. (2002) classifications of electrical connections according to their complexity (in terms of assembly). A rotary energy flow from a rotary-to-linear drive at 200 W is more difficult to alter than a hand powered linear translation energy flow.

Table 1 A modified list of functional basis

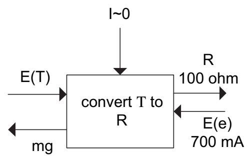
Flow category	Sub-category
Material	Human
	Gas
	Liquid
	Solid
Energy	Human
	Acoustic
	Electrical
	Electromagnetic
	Hydraulic
	Magnetic
	Mechanical
	Rot torque
	Translational speed
	Vibration
	Pneumatic
	Thermal
Information	Content
	Bandwidth

We analyzed flow difficulty through consultation with five system engineers, involved in the design and expert in the domains relevant to the injector head, and six system engineers involved in the sensor system design. We asked engineers for relative difficulty of different types of changes related to the products of their expertise. We asked questions about increasing and decreasing various flows by different percentages to find out if there is a difference in increasing or decreasing a flow value. We asked each engineer to estimate the person hours needed if a particular change was to be made. We also asked for the effect of each change on the existing processes to make sure that the change is in fact possible. Table 2 presents an example from the interview form used.

We generally find the following characteristics of any redesign difficulty. For small changes to the input or output characteristic, no redesign is required; the effort is zero, since the module is usually over-designed to some degree. At some level of change of input or output, redesign effort is required. Generally, over a useful domain of input and output, the redesign is scalable and often even linear with the input or output change. At a second level of input or output, however, a completely new module is required and massive redesign effort is required. This general behavior is graphed in Figure 2.

We mainly asked questions in the change range where the effect can be presumed somewhat linear i.e. above where the change is not already

Table 2 Part of the interview form used

Description	Black box	Change (everything else stays the same)	Person hours required for each change	Effect on existing processes none/low/med/high/impossible
Temperature sensor. Humidity transmitter. Inputs: power from larger circuit board via cable. T from the outside (range: −70 to +180 °C). Heat energy from remove water and remove HC (when not measuring). $I \sim 0$ is possible EM radiation getting through Outputs: R to the cable Fn = mg to structure	 I~0 E(T) convert T to R R 100 ohm E(e) 700 mA mg	Upper limit requiring no design change – stays within capacity:_____% input voltage increase	0	None
		Upper limit requiring no design change – stays within capacity:_____% input voltage decrease	0	None
		10% input voltage decrease (within capacity)		None
		30% input voltage increase		
		30% input voltage decrease		
		40% input voltage increase		
		40% input voltage decrease		
		100% input voltage increase		
		90% input voltage decrease		
		Lower limit that will require a completely new concept:_____% input voltage increase		
		Lower limit that will require a completely new concept:_____% input voltage decrease		
		Is there a common interaction with input voltage? What is it? What is the redesign effort for it?		

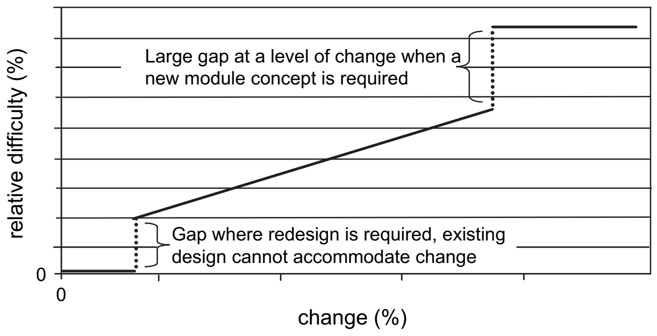


Figure 2 General redesign behaviour

designed in as over capacity of the original component. We also stayed where only modification is needed and not total redesign (Figure 2). This linearity was later supported by our data. We asked a few questions also in the far ends of the range as well about the built in over capacity of the modules to see how the results were affected by it, but closer examination of the far ends is not the focus of this study.

Making changes to a function in a function structure will often force changes to the functions surrounding it. Dealing with this ripple effect of a design change is what a design engineer must do before using our method, and estimate the extent of change of each function. The effort to change any function, though, is an independent estimate of our interviewees. Estimating design effort to change each function independent of the others is appropriate for two reasons: (1) Our goal is to be able to define modules that are easy to redesign in case there is a change somewhere in the system. For example, if there is a change in a module, we ask the design effort to accommodate that change while trying to maintain the remainder of the system unchanged. (2) In some cases, it is impossible to change a function without changing some of its neighbors. For example, changing a motor in power will require changing its controller. We left it to the designers (the interviewees) to back-propagate the changes as far as they felt was necessary. We only want to represent the redesign for each change by effort numbers, not the logic behind the back-chaining.

We obtained the estimated person hours required for each change from our interviews. We then normalized these results into relative hours compared to the original effort. For example, if the answer was 30%, it meant that to redesign the component to accommodate the specific change requires an additional 30% of the original design work. We also normalized the answers so that all interviewees had the same global average across all modules, to eliminate questionnaire biases between different interviewees. We did this separately for each case study since we did want to lose the company or product type specific content of the results.

We ran statistical tests of our regression analyses to ask whether the answers for each flow type came from different groups (cases) or not. This was done to see whether the two case studies should be kept separate or not. We did this by comparing the slopes of the regression curves of both case studies for each flow category. For flows for which we had the most data (electrical energy, bandwidth) the slopes were different with a confidence of 95%. For the others we have similar results but with a smaller confidence. Thus we decided to keep the cases separate. Then, to obtain the redesign effort metric, we fitted a regression line of form $y = b_1x + b_0$ for each flow type for both case studies. The slope of the line describes the difficulty of redesign. The slope is our redesign effort metric.

For flow types that occurred only once or twice in or products we only acquired a limited number of responses. If this was insufficient to fit a line, we approximated the redesign simply by calculating the relative change needed per 1% change from each response. We then averaged these to obtain the redesign score. We then repeated this for the flows for which we could fit a line and we concluded that an average is a good approximation. In earlier work (Holtta and Otto, 2003) we used only the latter, simpler, average model and thus the new results differ slightly from the earlier ones. In addition, we went back to the first case study companies and acquired additional data.

Finally, we grouped the responses, case by case, across all of the modules into acoustic, electrical, mechanical, thermal, and pneumatic energy, information, and material flows according to the functional basis. That is, for each of these categories we found the relative rework scores across all modules examined.

4 *Results*

Figure 3 shows an example of how the relative redesign effort changes as the redesign change percentage increases. The data points support a linear model. Another observation can be made from Figure 3; the data points seem to fan out as the percentage increases. This is an indication of the fact that it becomes harder for the engineer to estimate the required redesign effort as the change increases, since more options

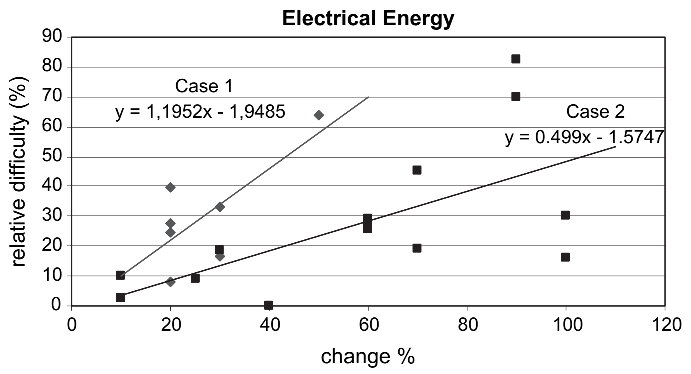


Figure 3 Linear regression for electrical energy flows. As expected, worse fit occurs at large redesign change levels

become available and the change becomes impossible to isolate to only a single module. The typical statistical approach to data that exhibits larger spread as the response grows is to transform the data by using a logarithmic or similar transformation. We chose not to do that here since, for the moment, we are primarily interested in smaller levels of design change typical to redesign, generally under 50%. Larger changes than that generally require architectural changes. Further, it was not always the case that transformed data improved the fit.

To change a flow by 1%, our results indicate what percentage additional design work is required. In general, we find that a 1% change in any product flow requires about 0–3% of the original design effort. Table 3 shows the additional design work, or the difficulty of change, for various types of the functional basis flow categories. We also included the R^2 and *p*-values to show how accurate each model is for cases where appropriate. As can be seen, the goodness of fit can vary substantially from under 0.5 to above 0.9. Non-linear equations will improve the fit on some of the flow types, but we present here the linear results.

In addition to the difficultly numbers themselves, also the goodness of fit statistics show an interesting result – some redesign difficulty levels are reasonably predictable, others are not. Mechanical energy modules are predictable at the case 1 companies, translational less so, and so backup plans should be considered when uncertainty.

The intended use of the table is twofold. First, it can help in preliminary conceptual architecture activities, to decide where to place module boundaries – better to place boundaries on flows with less difficulty. This is useful since typically a module is, or should be, developed by

Table 3 Relative rework needed as percentage of the original eort for dierent types of flows (*less than three data points) (increase only) (***average of sub-category scores)**

Flow category	Sub-category	Case 1			Case 2		
		Redesign effort	R^2	p	Redesign effort	R^2	p
Material	Solid	1.2*	–	–	–	–	–
	Gas	–	–	–	0.3*		
Energy	Acoustic	3.1*	–	–	–	–	–
	Electrical	1.2	0.55	0.058	0.5	0.46	0.06
	Mechanical						
	General***	1.0	0.37	0.111	–	–	–
	Rot torque	1.0	0.24	0.033	–	–	–
	Translation	1.0	0.49	0.189	–	–	–
	Pneumatic**	1.3	0.98	0.084	–	–	–
	Thermal	1.8*	–	–	0.3	0.85	0.003
Info	General***	0.8	0.33	0.066	1.0	0.75	0.027
	Content**	1.2*	–	–	1.7	1.00	0.000
	Bandwidth	0.4	0.33	0.066	0.2	0.49	0.054

a single team. It is easier to cope with complex interfaces within a team than across team or even company boundaries, when the other modules are developed by separate teams or outsourced. Second, it can help in assessing design changes to a product – if a module is increased or decreased in size, power, output speed, etc., how much design effort is required? This can help scope a project.

As we can see, the values are different for each flow type as well as for each case. This was expected. We hypothesized that different interactions have different effects in terms of redesign effort and we found this to be true.

We also expected that the values depend on the company and product in question. This is also true. For example, the case 1 companies had a redesign effort of 1.2 on electrical energy flows, whereas case 2 company had an effort of 0.5, less than half of case 1 companies. The second case study company is more confident about their abilities to adapt to change than companies in case study 1. Basically, they have more experienced engineers who can design more quickly.

A closer look at the table, however, reveals that even though the values are different across the two cases some generalizations can be made. For example, changing information flow bandwidth is considerably easier than changing information content. Also, electrical energy (typically voltage) is harder to change than information bandwidth. We believe this is due to the larger buffers typically used for bandwidth than for voltage. On the other hand, information content requires more design effort to change than an electrical flow.

The values in Table 3 are for a change, regardless of its direction since future changes are often unexpected. However, if one were to use the table for scoping a redesign project, the change of direction is known and one could use separate values for increasing and decreasing type of changes. Table 4 lists values for our two case studies. We left out categories for flows, for which we had data for the increasing value only or not enough data points for both increasing and decreasing the flow. The values in Table 4 cover the three basic flow types introduced first by Pahl and Beitz (1999) and commonly used by others. In general, we notice that decreasing a flow tends to require as much or less design effort than increasing it, but this is not always true.

For the first use, designing a flexible architecture use only the design effort complexity metric as such. We calculated a Pearson coefficient of approximately 0.9 for both case studies supporting the fact that the interface metric represents the difficulty of the interface, regardless of the design difficulty of the adjacent modules.

For the second use the numbers in Table 3 and Table 4 must not be used directly for scoping a redesign project, but must be multiplied by the original design effort as well as the change percentage. A large number in Table 3 or Table 4 may be favorable if the flow in the product is small. An illustrative example of this is the speaker and its acoustic energy flow in one of the injector heads. The design effort complexity metric value for acoustic energy is high (3.1) but in this product, a change in the acoustic energy means changing to a different speaker – selecting a new component from a catalogue. The original effort is mainly to decide

Table 4 Relative rework needed as percentage of the original effort for increasing and decreasing flows

Flow category	Sub-category	Case 1		Case 2	
		Redesign effort		Redesign effort	
		Flow increase	Flow decrease	Flow increase	Flow decrease
Energy	Electrical	1.5	1.0	0.4	0.8
	Mechanical	1.0	0.5	–	–
Info	Bandwidth	0.3	–	0.2	0.2

which speaker to choose from a selection of speakers, so the redesign effort is very high – close to that of the original. However, the selection and new documentation is a small effort in total, about 20 h as estimated by one of the engineers. Similarly a small design effort complexity metric value (1.2) for an electrical energy does not necessarily mean a trivial change in the product. For example, an engineer estimated the original design of the main control card in an injector head to be about 16 person weeks and about two person weeks for two indicator lamps. A 1.2% redesign effort is obviously much less for the lamps than for the control card. We also tested this statistically and we found no correlation between the original work effort and the design effort complexity number and therefore the product of the two should be used.

5 *Use*

To test our approach, we applied the procedure introduced above with the design effort complexity metrics to a gas sensor.

The gas sensor measures the gas content of an environment with a capacitor and a resistor. These values are transmitted to a circuit board where the actual gas content calculation is done. This information is used to control the sensing, the outside process, and to provide user with appropriate information.

5.1 *Build function structure*

We started by decomposing the product into smaller sub-systems down to the company assembly level. We developed the function interactions with material, energy, and information flows in between the function boxes. The function structure was validated with engineers involved in the design of the sensor to be sure that is was correct and that it met all the customer needs. A partial function structure of the gas sensor is shown in Figure 4.

5.2 *Modularize the product*

We chose to modularize the gas sensor using the DSM approach, specifically the algorithm by Thebeau (2001), since the goal in this case was to simplify interfaces and enable parallel development. The DSM identified modules based on the interaction between modules. Nine of the modules defined by the algorithm are shown using thick dashed lines in Figure 4. The first module involves the sensor head i.e. functions *protect sensor, align in space, absorb/release gas*, and *remove gas*. The second module includes the power supply (*on/off* and *supply power*) as well as the *control heating* function. The third module consists of functions *convert* ϵ_r to C and *transmit C value*. Two modules for a similar

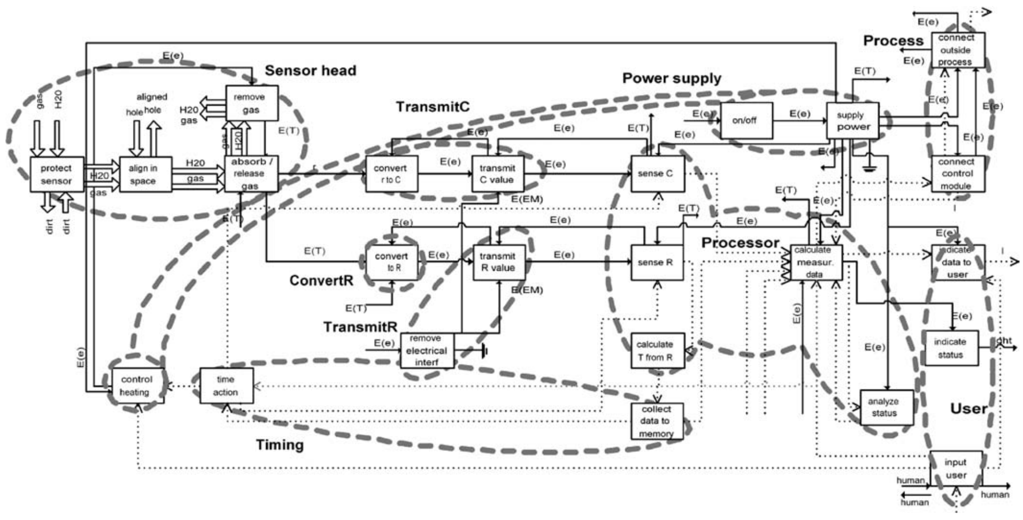


Figure 4 Partial gas sensor function structure with modules

parallel function chain were defined differently. One module is the single function *convert T to R* and the other consists of two functions *transmit R value* and *remove electrical interference*. The main controlling functions of the product were divided into two modules; one with the *time action* and *collect data to memory* functions and the other module is built around the *calculate measurement data* i.e. the processor. The last two modules connect the sensor to the process it is sensing (*connect control module* and *connect to outside process*) and to the user (*indicate data to user, indicate status*, and *input user*).

5.3 Calculate interface complexities

The gas sensor was one of the subjects of the second case study so we use the second values for the sensor's interfaces. We assign a design effort complexity number for each flow between modules. We sum up these values to come up with a total design effort complexity number for each interface. In addition we calculate the within-module design effort complexities to see if the interfaces within a module are simpler, thus better module boundary candidates, or more complex, and thus good to keep within a module. The total design effort complexity values are shown in Figure 5. The between module interfaces are underlined and the within module interfaces between functions are in *italic*.

5.4 Identify critical interfaces and define module boundaries

In this step we identified the module interfaces suggested by a DSM modularization method that require a considerable amount of redesign

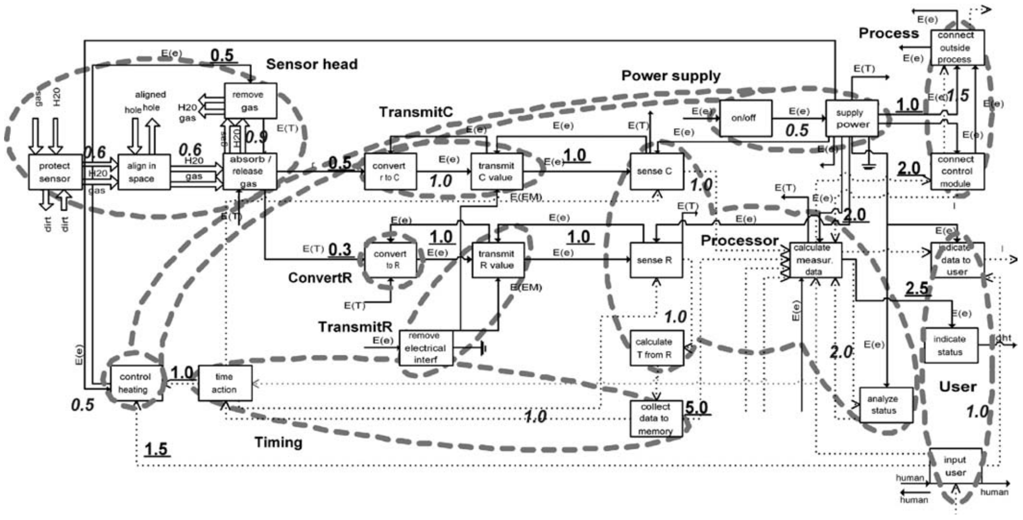


Figure 5 Partial gas sensor function structure with module interface design effort complexities

effort if changed. These interfaces are better kept within a module if possible and a simpler interface should be at the module boundary. This ensures that if there is a change to the module, the change causes minimal amount of rework to the rest of the system. Redesign ease, however, is only one criteria to be used to define module boundaries, so also other factors should be kept in mind during this step.

Looking at the gas sensor function structure in Figure 5 one interface can be clearly seen as a difficult interface; the interface between the *Timing*-module and the *Processor*-module. This interface consist of five information flows (each 1.0) between the functions in each module has therefore a design effort complexity score of 5.0. If anything should change in one of these modules, it would cause major design also in the other module. From the redesign ease point of view, these two modules should be kept together as a single module.

The sensor is a very software intensive product and thus the software causes many complex interfaces, for example the interfaces toward the user and the process. These interfaces however will be kept unchanged since the company offers a selection of these modules e.g. different displays.

The DSM algorithm combined the power supply with the *control heating* function. These two, however, share an interface of 0.5 between them and they could thus be easily separated. This would then also free the control heating function to be combined in a module with the time action module, for example, to avoid an interface of 1.0, or the function could be left as a single module since it does not have very critical interface to and adjacent function.

Finally, we summed up all the interfaces between the final modules to calculate a redesign ease score for the entire architecture. The score for the modularization suggested by the DSM is 19.3 (add underlined numbers in Figure 5). And if we combine the time action module into the calculate measurement data module and separate the control heating as a single function module, we get a score of 14.8 (see Table 5). The altered architecture is much better from the redesign point of view, but still in line with the simple interface a decoupled development goal.

One must be especially aware of not trying to combine functions into too large modules. In general, the modularization process is a balance between many factors. Other factors such as recyclability, variety, and reuse should be driving toward small modules. The idea of the architecture score is that now we can explore alternative architectures and then choose the best one in terms of several factors, minimizing design effort complexity being one. Further, for any selected architecture, the ease of design upgrades is well understood at the outset, and areas where one is uncertain or understands there will be future changes that can be accommodated.

Table 5 Interface design effort score for the alternative gas sensor architecture

From module	To module	Interface design effort score
Sensor head	Transmit C	0.5
Sensor head	Transmit R	0.3
Sensor head	Control heating	0.5
Power supply	Control heating	0.5
Power supply	Processor	2.0
Power supply	Process	1.0
Power supply	User	0.5
Control heating	Processor	1.0
Control heating	User	1.0
Transmit C	Processor	1.0
Convert R	Transmit R	1.0
Transmit R	Processor	1.0
Processor	Process	2.0
Processor	User	2.5
Total		14.8

Beyond architecting, the redesign percentages can also be used to roughly estimate the redesign needed if a specific change is made. This can be done for the whole architecture by multiplying the actual percent change of each flow by its relative redesign percentage at each interface and summing them all together.

The results shown here held true across three different corporate cultures, two types of products, and different modules. However, the scope of the analysis remained electro-mechanical devices. It is not clear how large a domain this can be expanded to while maintaining meaningful consistency of responses. There are several factors influencing the rework effort such as design process, representation of the design, tools used, etc. We did not consider variations in these but used the ones that the companies in this study deal with. Thus we cannot claim applicability of the metric values to all other domains, but we believe the same approach could be used elsewhere. As we showed with the two case studies it is extremely doubtful that a general single number coefficient that applies globally for all industries and product types can be derived. Each company or at least type of industry should determine the metric coefficient values of their own. However, the results do provide consistency and are effective for any company to construct and then use. That was our intention.

6 *Conclusions*

We introduced a method design module interfaces so that in case of a change the product is quick to adapt as well as to relatively compare proposed product architectures based on design effort complexity. Our method is an effort to help the definition of module boundaries after the module‘cores’ have been identified with a modularization method. We used the design structure matrix approach as an example and showed the use of our method through a real industrial case study. The design effort complexity metric developed also aids in evaluating modularized architectures. Our metric represents the difficulty that various module boundary interactions would have in terms of redesign effort. The interactions were represented by flows, mechanical, and electrical energy, information flows, etc., into- and out-of-a-function. We found that to change a flow by 1%, approximately 0−3% more design effort is required to accommodate the change, depending on the interaction type. We also found that decreasing a flow value tends to cause, in general, as much or less rework than increasing a value.

The results on design difficulty to change a flow were interesting in and of themselves for understanding how hard it is to change the interfaces for various types of flows. We found their use in modularity makes for much improved system architecting. For any proposed module within a product architecture, the interconnection with the rest of the product could be easily assessed for design change difficulty and a product could be designed to be easy to change in the first place.

Acknowledgements

The authors would like to thank the MIT Center for Innovation in Product Development and Helsinki University of Technology, Department of Machine Design for supporting this research. We finally thank the interviewees at all case study companies, Hannu Valo for help with case study 2, and Product Genesis, Inc.

References

Bass, L, Clements, P and Kazman, R (2003) *Software architecture in practice* (2nd edn) Addison-Wesley

Blackenfelt, M (2000) *Managing complexity by product modularization*, Ph.D. thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm

Boothroyd, G, Dewhurst, P and Knight, W (2002) *Product design for manufacture and assembly* (2nd edn) Marcel Dekker Inc, New York

Braha, D (2002) Partitioning tasks to product development teams *ASME Design Engineering Technical Conferences* Montreal

Braha, D and Maimon, O (1998) The measurement of a design structural and functional complexity *IEEE Transactions on Systems Man and Cybernetics Part A* Vol 28 No 4 pp 527–535

Camuffo, A (2001) Rolling out a "World Car": globalization, outsourcing and modularity in the auto industry, Working Paper, International Motor Vehicle Program, Massachusetts Institute of Technology

El-Haik, B and Yang, K (1999) The components of complexity in engineering design *IIE Transactions* Vol 31 pp 925–934

Ericsson, A and Erixon, G (1999) *Controlling design variants: modular product platforms* ASME press, New York

Fixson, S (2001) Methodology development: analyzing product architecture implications on supply chain cost dynamics, Paper Presented at the 5th Conference on *Technology, Policy, and Innovation "Critical Infrastructures*", Delft, The Netherlands

Gonzalez-Zugasti, J P, Otto, K N and Baker, J D (2001) Assessing value in platformed product value design *Journal of Research in Engineering Design* Vol 13 pp 30–41

Gupta, S and Krishnan, V (1999) Integrated component and supplier selection for a product family *Production and Operations Management* Vol 8 No 2 pp 163–181

Hirtz, J, Stone, R B and McAdams, D A (2002) A functional basis for engineering design: reconciling and evolving previous efforts *Journal of Research in Engineering Design* Vol 12 pp 65–82

Holtta, K and Otto, K (2003) Incorporating design complexity measures in architectural assessment *ASME Design Engineering Technical Conferences* Chicago

Holtta, K and Salonen, M (2003) Comparing three modularity methods *ASME Design Engineering Technical Conferences* Chicago

Krishnan, V and Gupta, S (2001) Appropriateness and impact of platform-based product development *Management Science* Vol 47 No 1 pp 52–68

Kurfman, M, Stock, M E, Stone, R, Rajan, J and Wood, K (2003) Experimental studies assessing the repeatability of a functional modeling derivation method *Journal of Mechanical Design* Vol 125 pp 682–693

Maier, M W and Rechtin, E (2000) *The art of systems architecting* (2nd edn) CRC Press

Mikkola, J (2000) Modularity, outsourcing, and inter-firm learning *DRUID Summer conference* Rebild, Denmark

Otto, K (2001) A process for modularizing product families *International Conference on Engineering Design* Glasgow

Otto, K and Wood, K (2001) *Product design: techniques in reverse engineering and new product development* Prentice Hall, Upper Saddle River, NJ

Pahl, G and Beitz, W (1999) *Engineering design* (2nd edn) Springer-Verlag, London

Parnas, D L (1972) On the criteria to be used in decomposing systems into modules *Communications of the ACM* Vol 15 pp 1053–1058

Pimmler, T and Eppinger, S (1994) Integration analysis of product decompositions *ASME Design Engineering Technical Conferences* Minneapolis

Smith, J S, Robb, M D, Duffy, A H B, Thomson, A and Nisbet, C (2001) An experience of modularity through design *International Conference on Engineering Design* Glasgow

Sosa, M E, Eppinger, S D and Rowles, G M (2000) Understanding the effects of product architecture on technical communication in product development organizations, MIT Sloan School of Management, Working paper # 4130.

Stake, R B and Blackenfelt, M (1998) Modularity in use – experiences from five companies, *4th WDK Workshop on Product Structuring Delft*, The Netherlands

Stone, R B, Wood, K L and Crawford, R H (2000) A heuristic method for identifying modules for product architecture *Design Studies* Vol 21 No 1 pp 5–31

Suh, N (2001) *Axiomatic design: advances and applications* Oxford University Press, New York

Thebeau, R E (2001) *Knowledge management of system interfaces and interactions for product development processes*, Masters thesis, System Design & Management Program, Massachusetts Institute of Technology

Ulrich, K (1995) The role of product architecture in the manufacturing firm *Research Policy* Vol 24 pp 419–440

Ulrich, K T and Eppinger, S D (2004) *Product design and development* (3rd edn) McGraw-Hill/Irwin, New York

Whitney, D E (2004) *Mechanical assemblies: their design, manufacture, and role in product development* Oxford University Press