Publication IV

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

Proceedings of DETC'03
ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference
Chicago, Illinois USA, September 2-6, 2003

DETC2003/DTM-48648

INCORPORATING DESIGN COMPLEXITY MEASURES IN ARCHITECTURAL ASSESSMENT

Katja M. M. Holtta
Helsinki University of Technology
Department of Machine Design
P.O. Box 4100
02015 HUT
Finland
Tel +358 9 451 5072
Fax +358 9 451 3549
Email: Katja.Holtta@hut.fi
and
MIT Center for Innovation in Product Development

Kevin N. Otto
Product Genesis, Inc.
245 Bent St
Cambridge, Massachusetts 02141
USA
Tel +1 617 234 0070
Fax +1 617 354 8304
Email: Kevin_Otto@ProductGenesis.com

ABSTRACT

A feature of good modularity is the ease of changing a module within a product. Existing modularity methods use subjective or qualitative attributes to evaluate architectures. We develop a method to relatively compare proposed product architectures according to design complexity. Our metric represents the difficulty that different module boundary interactions, represented by flows in and out of a function, would have in terms of redesign effort. We decomposed medical injector head systems and conducted interviews in two companies to find out a relative redesign effort for various interaction types, e.g. electrical and mechanical connection, signal flows, etc. We found that to change a flow by 1%, 1-4% more design effort is required, depending on the interaction type. We also found that decreasing a flow value causes, in general, less rework than increasing a flow. Our metric proved to be a valuable tool in estimating the redesign difficulty of an architecture.

KEYWORDS: product architecture, modularity, complexity, redesign

INTRODUCTION

A module, as defined in this paper, is a structurally independent building block of a larger system with well-defined interfaces. It is fairly loosely connected to the rest of the system allowing an independent development of the module as long as the interconnections at the interfaces are well thought of. [1, 2]. The collection of modules and their interconnectivity define the product architecture for a product. One advantage of modularity is the reduced effort required to redesign aspects of a design for such purposes as technology upgrades or styling changes. In this paper we develop a method to define module boundaries and relatively compare proposed product architectures on design complexity, as measured by design difficulty to change the interfacing flows between modules. This metric is not intended to be a sole determining factor to modularize a product, but rather, is one of several metrics a design engineer would use to consider alternative modularizations. We provide here a well defined quantitative exploration of the design effort complexity metric.

This work falls within the efforts to improve understanding of developing modular products. The advantages of modularity are well known as the possible economies of scale and scope such as in parts sourcing [2, 3]. Modularity provides flexibility that enables product variations and technology development without changes to the overall design [1]. The same flexibility also allows for independent development of modules, which is useful in concurrent or overlapped product development activities [4], collaborative projects, or when buying the module from a supplier [5]. Modularity eases the management of complex product architectures [1] and therefore also their development. Modularity can also be used to create product families [6]. This saves design and testing costs and can allow for greater breadth of design options, though one must be aware of possible excess functionality costs if a low cost and low functionality product is instantiated with a high cost module in order to use the same module in several products [7, 8].

One feature of good product modularity is the ease with which modules can be changed within a product, their degree of isolation. Yet, there are few methods to quantify modularity and to thereby choose module boundaries. Stone *et al.* developed a heuristic method to identify modules by finding the dominant flow, branching flows, or conversion-transmission

function pairs within a function structure [9]. Ericsson, on the other hand, developed modular function deployment that groups functions (or components) according to strategic aspects such as technology evolution, planned changes, or styling [1].

Blackenfelt combines Ericsson's method with the design structure matrix (DSM) to cluster technical solutions into modules based on their interaction strengths [10], where the ratings are estimates. Further, Kota et al. present a benchmark method to compare one's own platform to a competitor's platform. The method considers manufacturing, component's size, and material but it is not a platforming tool. [11]

Current modularization methods help identify module "cores" i.e. functions around which a module is built. The exact module boundary definitions, however, are left up to the designer, particularly with respect to considering where to place the boundaries and the impacts of such decisions. Engineers need decision support with such preliminary design activity, particularly here in grouping functionality into modules [12]. Ericsson, in his modular function deployment [1] recognizes the importance of a step for interface design, but his 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.

To this end, modularity assessment must include many factors, such as ease of upgrades, ease of supporting variants, design ease, supplier capability [13], and manufacturing support, for example. Among these, one important factor is design complexity, the ease with which a module design can be redesigned without impacting its interface and the rest of the product. In some industries such as aerospace, electronics, and the high tech medical industries, design effort can become a dominant design criterion in architectural assessment.

Literature on design complexity primarily considers design process modeling and design process complexity. For example, Braha and Maimon [14] 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. Suh [15] 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. El-Haik and Yang [16] discuss further 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 define module boundaries, the ideal interfaces (in terms of design effort) between modules; the main goal is not a uniform number to represent all criteria upon which to evaluate the complexity of a design, but rather we focus on 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.

Blackenfelt [10] describes complexity in context of modularity with the number and type of relations and elements in a product, not design difficulty. Also Maier and Rechtin [17] describe architecture complexity by the amount of connections, or communication in case of software, between modules. In a given architecture, the number and type of elements is given, but the number and type of relations can be affected. Blackenfelt's complexity metric, similar to others, treats all of these relations as having the same difficulty, which is not generally the case. There is 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. A rotating axle must cause more design problems if there is a change than a stationary mechanical connection, such as a bolt. For example, which is more difficult to compensate, a 30% increase in signal bandwidth or a 30% increase in operating voltage? As we will demonstrate, some interaction types are more difficult to modify than others. In this paper we develop a method to define module boundaries and relatively compare proposed product architectures on design complexity, as measured by design difficulty to change the interfacing flows between modules.

One should notice that our metric alone is not meant for deciding the number or size of the modules. Our metric along with others, such as assemblability, cost, supplies, etc. are all important criteria to use in such a multi-criterion decision. We provide a structured means to represent one metric – design effort complexity – in such a decision. There are other approaches to estimate the number of modules one should use in a design. For example, Ericsson [1] simply develops the ideal number of modules as approximately the square root of the number of parts to be assembled. To determine the size of a module one could also refer to Braha [18] who suggests using the connection of product development teams and tasks to product modules. 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.

The remainder of the paper is structures as follows. We develop the general step-by-step approach that a user would use in the following section. We then explain the research methodology we used and specific quantitative results. The subsequent section then presents specific interesting results of our study. The Use section then shows how to use our design complexity metric via an example. We end the paper with conclusions.

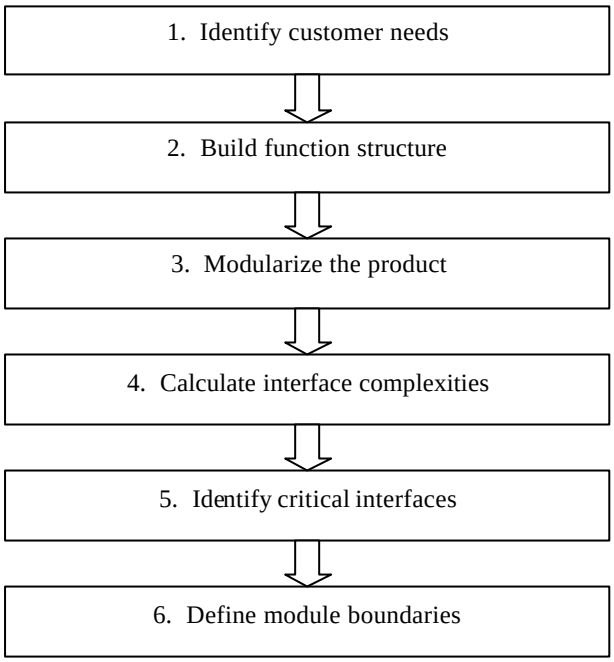


Figure 1 The procedure.

APPROACH

Our approach consists of a six steps procedure to modularize a product and for choosing and evaluating module boundaries. We do this by constructing a measure of the design complexity of the module boundaries in terms of design effort to change the boundaries. We start by identifying the customer needs, use them to build a function structure, and then move on to modularize the product. We will then calculate a design complexity metric for each interface within each module and at their boundaries. The next step is to identify the most critical interfaces and reorganize the module boundaries. Figure 1 shows the basic structure of the procedure.

Note that the same procedure can be used independent of whether the product exists already or not. For a new product concept one can follow the method developed by Otto and Wood [6] using function structures for steps 1 and 2. They start similar to our procedure by identifying customer needs and transforming them into function chains that are weighted with the importance to the customer. The most important functions are drawn first, they are combined into parallel function chains, and then the additional functions are added to form a complete function structure of the product. In case of an existing product, 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.

In the third step, modularize the product, one can choose from many methods introduced before. A function structure is a useful systems engineering diagram upon which modularity questions can be graphically posed and be well understood by a practicing design team. The function structure can be used with any of the modularity methods even though they are all not based directly on the function structure. We choose here to use the modularity heuristics developed by Stone *et al* [9].

For the fourth step, calculate interface complexities, we have developed an approach to evaluate the design complexity of an interface. We use the function structure flows to represent interactions between functions. Fixson suggests that interactions have different intensities [19]. 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 engineering effort, measured using estimates of required hours. The interface complexity metric can be calculated for various flow types, as will be shown in the results section.

The calculations can be used to identify critical interfaces in a product architecture for the fifth step. The larger the design complexity metric on a specific interface, the better it is to keep the interface within a module. And similarly, the smaller the design complexity metric at an interface, the better candidate the interface is to be at a module boundary.

The sixth and final step is to define the module boundaries. Existing modularity methods generally do not give definite choices for module boundaries, but suggestions for module "cores". Our design complexity metric is one measure to be factored in with the many other metrics one must consider when establishing boundaries. These include supplier capability of module complexity, assembly costs, serviceability and maintenance, and many others. Our design complexity metric can help choose the best alternative from various modularization schemes in terms of minimizing the design effort at an interface.

METHODOLOGY

Our approach to constructing design effort indices for any flow is to examine many modules and ask 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. We did this with several engineers at two different firms and dozens of modules. The posed questions themselves, however, were not so straightforward to extract useful data from. Many factors can impact design time, notably experience level, familiarity, and overhead of any particular corporate culture.

We accounted for these sources of error in two ways. First, we only worked with highly experienced system engineers that did design work on different design variants of the injectors 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. Third, we normalized our redesign results against the hours to complete the original baseline design. Relative estimates proved more repeatable than absolute hour estimates. Further, for our purposes, relative comparisons of design effort complexity is sufficient for comparing design concepts, actual hours estimates have unnecessary resolution for preliminary architectural alternative assessments.

The product domain we explored was electro-mechanical medical devices. We chose this since they are highly

engineered, amenable to a function structure systems representation, and design experts were available. We decomposed two slightly different injector heads used to inject contrast into a patient's bloodstream during imaging procedures. We first built function structures for the product architectures. We decomposed the product to the assembly level of the manufacturer. We also assigned each component or sub-system a function e.g. a motor was also named "convert electricity into rotation". We then represented all the connections between the components or sub-systems with 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 [6], 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 structure was complete, we took each of the functions / sub-systems represented by each block in the diagram, and analyzed the relative redesign effort for each flow entering and exiting the sub-system. Each function in our structure is one-to-one with an identifiable physical subset of the product.

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*. [20] (see Table 1). We used energy flows of electrical, mechanical, and pneumatic, information flows, 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 a signal 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 two 24 V electrical flows going to the functions *emit light 1* and *emit light 2*. In a sense these are information flows telling the lamps which one should be lit according to the current status of the injector. In practice, however, it is a simple on/off connection and the intelligence, or the information content, is in the preceding function. Thus the two flows are represented as electrical flows.

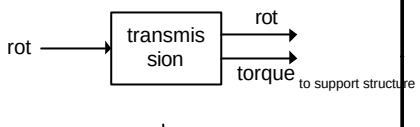
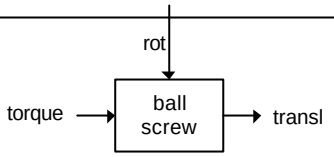
Table 1 A modified list of functional basis.

Flow category	Sub-category
Material	Human
	Gas
	Liquid
	Solid
Energy	Human
	Acoustic
	Electr voltage
	Electromagnetic
	Hydraulic
	Magnetic
	Mechanical Rot torque Transl speed Vibration
	Pneumatic
Signal	Content
	Bandwidth

Our hypothesis is that this classification serves as an indicator of interface difficulty. Each flow type is parameterized with a small number of descriptors, e.g., an information flow of 48 connectors at high bandwidth is more difficult to alter than an information flow of 2 wires, on/off. This is similar to Boothroyd and Dewhurst's classifications of electrical connections according to their complexity (in terms of assembly) [21 p.153]. A rotary energy flow from a rotary-to-linear drive at 175 W is more difficult to alter than a hand powered linear translation energy flow.

We analyzed flow difficulty through consultation with five system engineers, involved in the design and expert in the domains relevant to the injector head. We asked engineers for relative difficulty of different types of changes. 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 also asked for the effect of each change on the existing processes to make sure that the change is in fact possible. Following is a table (Table 2) presenting an example from the interview form used. 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 designed in as over capacity of the original component and no change is needed and below where only modification is needed and not total redesign. We did ask a few questions also in the far ends of the range to see how the results were different there, but closer examination of the far ends is left for future studies.

Table 2 Part of the interview form used.

Description	**Black Box**	**No of attachm. surfaces**		**Change** (everything else stays the same)	**Man hrs req f each change**	**Effect on existing processes** none / low / med / high / impossible
		Part attaches to	Attached to the part			
Transmission. Input: Rotation from the motor axle Output: Rotation to the ball screw				input torque increase 30%		
				input torque decrease 30%		
Ball screw. Inputs: torque from transmission, human control knob Outputs: linear movement to move the plunger				Input torque increase 30%		
				Transl speed change 20%		
				Transl stroke increase 20%		

Making changes to a function in a function structure will often force changes to the functions surrounding it. Dealing with this chaining impact 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 three reasons: (1) Incorporating the system changes would make it seem that no matter which component in the system is changed, the redesign effort is always the same, since every component in the system changes. (2) 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 the system changes outside a module, we ask the design effort to accommodate that change while maintaining the remainder of the system unchanged. (3) 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.

From our interviews, we obtained the estimated man hours required for each change. 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 calculated relative rework needed compared to the original effort for each one percent change in a flow by averaging all answers for each type of change and dividing it by the percent change in the questionnaire (Eq. 1). We also calculated standard deviations.

$$\text{Rework}_{1\%\,\text{change}} = \frac{\text{average}}{\text{percent change}} * 100 \qquad (1)$$

We then grouped the answers across all of the modules examined into electrical, mechanical, and pneumatic energy, information, and material flows according to the functional basis. That is, for each of these categories we found the average relative rework percentage across all modules examined. Similar to the way we posed the original questions, we calculated both the rework for increasing or decreasing a flow value as well as rework needed to change a flow regardless of the direction.

RESULTS

To change a flow by one percent, our results indicate what percentage additional design work is required. In general, we find that a 1% change in any product flow requires about 1-2% 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.

The intended use of the table is twofold. First, 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. Second, it can help in preliminary conceptual architecture activities, to decide where to place module boundaries – better to place boundaries on flows with less difficulty.

Note that the numbers in Table 3 must not be used directly for these comparisons, but must be multiplied by the original design effort. Large number in Table 3 may be favorable if the flow in the product is small. An illustrative example of this is the *emit acoustic vibration* function, a speaker, and the acoustic energy flow in the injector head. The design effort complexity metric value for acoustic energy is high (3.8%) 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 which speaker to choose from a selection of speakers, so the redesign effort is close to that of the original. However, the total selection and new documentation is a small effort in total, about 20 hours as estimated by one of the engineers, compared to the total design effort of the whole product. 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 control injector head, main control card, to be about 16 man weeks and about 2 man weeks for the emit light functions, injector arm status indicator lamps. A 1.2% redesign effort is obviously much less for the lamps than for the control card.

Table 3 Relative rework needed as percentage of the original effort for different types of flows. (with standard deviations)

Flow category	Sub-category	Difficulty of change
Material	Solid	1.1% (±1.3%)
Energy	Acoustic	3.8% (±5.2%)
	Electr voltage	1.2% (±0.5%)
	Mechanical	
	General	1.7% (±1.3%)
	Rot torque	1.7% (±1.4%)
	Transl speed	1.0% (±0.4%)
	Pneumatic	3.2% (±1.5%)
	Thermal	2.2% (±1.7%)
Signal	General	1.3% (±0.6%)
	Content	1.4% (±0.3%)
Spatial	Bandwidth	1.3% (±0.9%)
	Transl stroke	1.5% (±0.2%)

Beyond this direct application, there are some interesting insights to be gained from the results. For example, contrary to many beliefs (including at the participating companies) a signal flow (software) change is not necessarily easier than a mechanical change nor is mechanical change necessarily any easier than a software change.

There are similar insights within each flow type. In general it is always easier to decrease flow levels than increase them, as one might expect (see Table 4). For example, with mechanical rotational energy flows, a torque x speed increase by 1% requires 1.8% more work, whereas torque x speed decrease needs only 0.9% more work, a factor of 2 difference. In other words, changing torque from 15 Nm to 20 Nm requires approximately 54% of additional work relative to the original design hours where as changing torque from 15 Nm to 10 Nm requires only approximately 27% of rework. More pronounced, a pressure increase of 1% requires 4.2% more work, whereas a pressure decrease requires 2.2% more work. In other words, a 10% change in pressure causes 42% additional design work if the pressure is increased whereas decreasing pressure by the same 10% causes only 22% more work. Some energy flow types are more difficult to accommodate than others.

Also of note, energy flows are interesting in their units of power – energy and speed. Often, the change in one of these factors is important, and the design difficulty is different for two components. For example, consider changing torque or speed compared to changing power = torque x speed. Each of theses scenarios has different design change difficulties, as shown in Table 4.

Table 4 Added design difficulty to increase and decrease the interface for various flow changes by 1% in descending order.

Flow interface	Difficulty	
	Increase	Decrease
Pressure (Const flow)	4.2%	2.2%
Torque x Speed	1.8%	0.9%
Torque (Const speed)	2.9%	1.0%
Bandwidth	1.4%	0.6%
Voltage (Const I)	1.2%	1.3%
Speed (Const torque)	1.9%	0.6%

USE

To demonstrate and validate our approach, we applied the procedure introduced above with the design complexity metrics to a product, again an injector head, keeping the design domain within the family of injector heads. We started by decomposing the product into smaller sub-systems down to the company assembly level. We then assigned each component a function using the functional basis [19] vocabulary. We developed the function interactions with material, energy, and information flows in between the function boxes. The final function structure of the injector head is in Fig. 2.

We chose to modularize the injector head using the function structure heuristic approach [9]. We found the function chain *convert electricity to rotation – transmit torque – decrease speed – convert rotation into translation* as the dominant flow. *Convert human force to rotation*, *sense syringe*, the two *syringe size* sensors, *emit acoustic vibration*, *indicate data*, *import user data*, the two *emit light* functions, and *connect syringe* form modules candidates according to the branching flow heuristic. Functions *convert human force to rotation* and *convert rotation into translation* as well as function pair *convert electricity to rotation – transmit torque* are identified by the conversion-transmission pair heuristic.

As is inherent to the function structure heuristic methods, the module choices overlap and are not definite by the heuristics themselves. The chosen function chains could easily be a function shorter or longer without breaking the heuristic rules. For example the dominant flow could as well include functions *change syringe volume* and *store contrast* or not. The heuristic method simply provides suggestions, and it is up to the designer to choose among these suggestions based upon judgment. Here we provide quantification of one metric, design complexity in terms of redesign effort.

The function structure includes electrical and mechanical energy, information, and material types of flow interfaces. The design complexity difficulties are assigned for each flow and summed at each interface. Figure 2 shows the summed interface complexities at each interface.

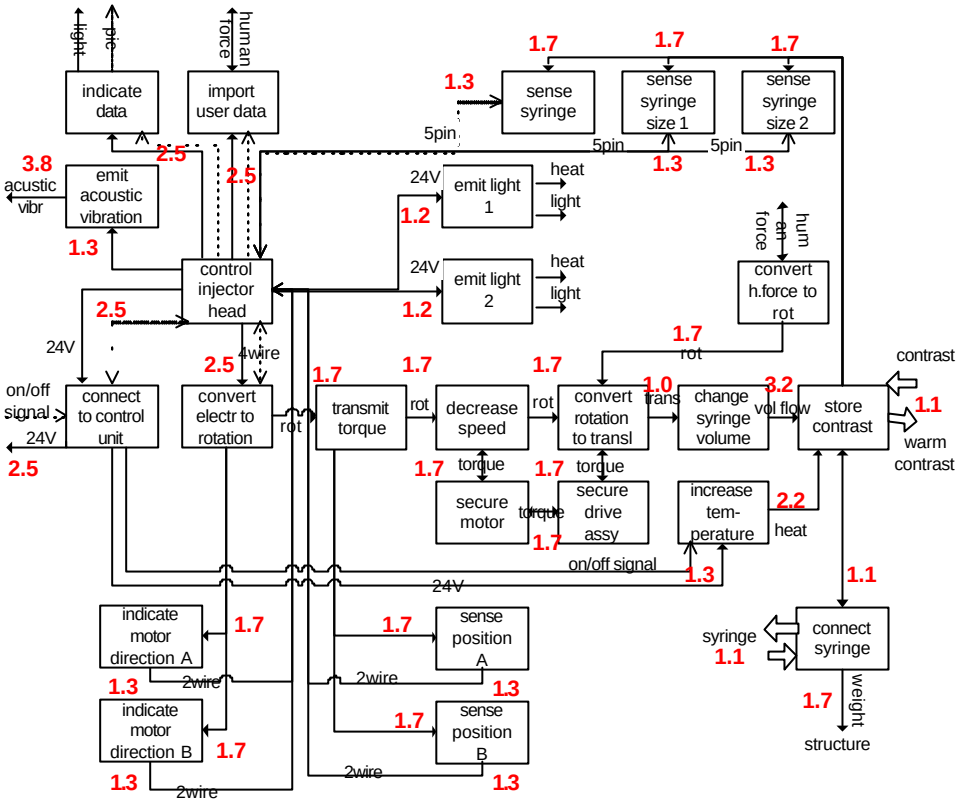


Figure 2 An injector head function structure with interface complexities.

It is now easy to identify the most critical interfaces. The most complex interfaces are ones with *acoustic vibration* by the function *emit acoustic vibration* and *volume flow* from function *change syringe volume* to function *store contrast*. These interfaces should be paid extra attention to. For example the volume flow interface should not be a module interface but inside a module. Thus we decide to lengthen the dominant flow module to *convert electricity to rotation – transmit torque – decrease speed – convert rotation into translation - change syringe volume – store contrast*. Another choice would be to use the short dominant flow and combine the two functions around the critical volume flow interface into a separate module. This calculated result proved well within agreement of our system design engineering experts.

Another example of the use of the design complexity metric is to decide how to group the modules not identified by any of the module heuristics. Say we want to decide whether to include the position sensing (potentiometers) into a module with either the *control injector head* function (main control board in the injector head) or with the drive system i.e. the dominant flow function chain. The percent change at the interface to the *control injector head* is 1.3% and at the interface to the drive 1.7%. This suggests that in order to minimize the needed rework, it is better to include the position sensing function to the drive to avoid the more complex interface and thus more potential redesign work. Again, this calculated result proved in agreement with our design engineer experts.

We group the rest of the functions using the same principal of minimizing the design complexity value at the module boundaries. We can now calculate the total design effort complexity of the architecture. We do this by summing up all the design complexity values at the module interfaces. The idea is that we can now explore alternative architectures and then choose the best one in terms of several factors, minimizing design 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 can be accommodated.

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 two vastly different corporate cultures and different modules. However, the scope of the analysis remained electro-mechanical medical devices, indeed specifically injector heads. 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. It is not clear and 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.

CONCLUSIONS

We introduced a method to relatively compare proposed product architectures based on design complexity in terms of redesign effort. Our method is an effort to help the final defining of module boundaries after the module "cores" have been identified with a modularization method. We used the function structure heuristics approach as an example. The 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 are represented by flows, mechanical and electrical energy, signal flows, etc., in and out of a function. We found that to change a flow by 1%, approximately 1-4% more design effort is required to accommodate the change, depending on the interaction type. We also found that decreasing a flow value causes, in general, less rework than increasing a value.

The results on design difficulty to change a flow are interesting in and of themselves for understanding how hard it is to change the interfaces for various types of flows. We find 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 can be easily assessed for design change difficulty.

ACKNOWLEDGMENTS

The authors would like to than the MIT Center for Innovation in Product Development and Helsinki University of Technology Department of Machine Design for funding this research. We would also like to give or appreciation to Thomas Roemer and Victor Tang for valuable insights. We finally thank the interviewees at both companies and Product Genesis, Inc.

REFERENCES

1. Ericsson, A. and Erixon, G., 1999, *Controlling design variants: Modular product platforms*, ASME press, New York, NY.
2. Baldwin, C.Y. and Clark, K.B., 2000, *Design rules. Volume 1. the power of modularity*, The MIT Press, Cambridge, MA.
3. Stake, R. B. and Blackenfelt, M., 1998, "Modularity in use – experiences from five companies", *4th WDK Workshop on Product Structuring*, Delft, The Netherlands.
4. Roemer, T.A., Ahmadi, R., and Wang, R.H., 2000, "Time-cost trade-offs in overlapped product development", Operations Research, **48** (6), pp.858-865.
5. 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.
6. Otto, K. and Wood, K., 2001, *Product design: techniques in reverse engineering and new product development*, Prentice Hall, Upper Saddle River, NJ.
7. Gupta, S. and Krishnan, V., 1999, "Integrated component and supplier selection for a product family", Production and Operations Management, **8** (2), pp.163-181.
8. Krishnan, V. and Gupta, S., 2001, "Appropriateness and impact of platform-based product development", Management Science, **47** (1), pp.52-68.
9. Stone, R. B., Wood, K. L. and Crawford, R. H., 2000, "A heuristic method for Identifying Modules for Product Architecture", Design Studies, **21**, Issue 1, pp. 5-31.
10. Blackenfelt, M., 2001, *Managing complexity by product modularization*, Ph.D. thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm.
11. Kota, S., Sethuraman, K., and Miller, R., 2000, "A Metric for Evaluating Design Commonality in Product", Journal of Mechanical Design, 122, pp. 403 – 410.
12. Smith, J. S., Robb, M. D., Duffy, A. H. B., Thomson, A. and Nisbet, C., 2001, "An experience of modularity through design", Proc., *International Conference on Engineering Design*, Glasgow, pp. 467-474.
13. Mikkola, J., 2000, "Modularity, outsourcing, and inter-firm learning", *DRUID Summer conference 2000*, Rebild. Denmark.
14. Braha, D. and Maimon, O., 1998, "The measurement of a design structural and functional complexity", IEEE Transactions on systems, man, and cybemetics, Part A, **28**, No 4, pp 527-535.
15. Suh, N., 2001, *Axiomatic Design: Advances and Applications*, Oxford University Press, New York, NY.
16. El-Haik, B., Yang, K., 1999, "The components of complexity in engineering design", IIE Transactions, **31**, pp. 925-934.
17. Maier, M. W. and Rechtin, E., 2000, *The art of systems architecting*, 2nd ed., CRC Press.
18. Braha, D., 2002,"Partitioning tasks to product development teams", Proc. of *ASME 2002 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Montreal, Canada.
19. 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.
20. Hirtz, J., Stone, R. B., and McAdams, D. A., 2002, "A functional basis for engineering design: Reconciling and evolving previous efforts", Research in Engineering Design, 12, pp. 65-82.
21. Boothroyd, G., Dewhurst, P., and Knight, W., 2002, *Product Design for Manufacture and Assembly*, 2nd ed, Marcel Dekker, Inc., New York, NY.

