DETC2009-86872

INTERFACES PER MODULE IS THERE AN IDEAL NUMBER?

Andrea Dobberfuhl
Modular Management USA
Bloomington, MN, USA

Mark W. Lange, Ph.D.
Modular Management USA
Bloomington, MN, USA

ABSTRACT

Identifying an appropriate level of abstraction for a technical system is a task often left to more experienced product designers. Based on a systematic approach to modular product design, experience has shown that module interfaces are a key factor in identifying this appropriate level. The foundation for this research is that there are an ideal number of interfaces that corresponds to three semiotic, or signed, notational levels of product design communication. This report will present an empirical metric that represents this ideal number, which can be used to assess the quality of the decomposition based on the number of interface types. The conclusion of the report is that the empirical value very closely matches the semiotic theory of three interfaces per module.

NOMENCLATURE

<i>DFA</i>	=Design for Assembly
<i>IM</i>	=Interface Matrix
<i>MFD</i>	=Modular Functional Deployment
<i>QA</i>	=Quality Assurance
<i>QFD</i>	=Quality Function Deployment
<i>VoC</i>	=Voice of the Customer
<i>US</i>	=United States

INTRODUCTION

The decomposition of a product into functional blocks or modules is an engineering strategy for reducing the complexity of a design problem. Systematic modular architecturing is one approach applied to achieve a strategically viable result, which allows the description of a product in terms of the modules, interfaces between these modules and the rules used to describe the possible combinations. For the designer who is executing

this process, there are decision points based on personal intuition and process guidelines. One intuitional decision point is reached when the appropriate level of granularity for the content of the functional modules is met. At this time the visible content of interfacing [1] is at a reasonable and manageable level. This can be represented as the number of interfaces between modules.

We propose that to help someone who is working with the design of a modular system, who has not yet gained the intuition, it would be useful to use a metric or measure to guide them in their decision. It is the intent of this article to introduce the development of this metric guided by the following questions;

- Are there an ideal number of interfaces for a resulting module concept?
- Does this number remain consistent irrespective of the scope of the module system or number of modules?
- Does the number of interfaces support a semiotic theory of product design communication for interface levels?

BACKGROUND

Product architecture is the physical building blocks of a product defined by what they do and what other parts of the device it interfaces with [2]. One attribute of product architecture is the building blocks or “chunks” contain one or a few functions in their entirety. The second attribute is that the interactions or interfaces between chunks are well defined and are typically fundamental to the primary function of the product. Product architecture allows for simultaneous design and testing on the building blocks over the development of the product. It also allows a company to offer differentiated products to their customers while utilizing a set of shared

components governed by a product platform. Product architectures can be created for any product ranging from software to passenger trains. The focus of this research will be on products that are physical, discrete, and engineered.

Product architecting using modules, interfaces and configuration rules, has been cited in key articles [3] [4] and books [1] [2] as a successful strategic approach for product management. Yet the implementations of such an approach are not fully understood. For example, in a study conducted on the interactions between engine components [5], found that creating a “more-modular” architecture improved technical communications between design teams. This improvement was evidenced in fewer interactions being missed and more predictable workloads, even if they found that too much focus on modularity led to a narrow sighted view at the subcomponent level. In another study on interface management [4], found that interface management is a useful means of moving from single product to product family development, which requires very deliberate effort in order to realize the benefits. The research reported in this article will not address the strategic or management importance of product architecting, instead our focus is on the design engineer’s act of synthesizing a product architecture.

It is widely understood that there are two types of product architectures, integral and modular [2]. An integral architecture has one function that is contained in a few chunks or many functions in a single chunk. The interfaces between integral chunks are clearly dependent on the technical solutions used to realize the product’s primary function, making this means of interfacing very rigid for configuration of product variants. In modular architecture, a chunk has one or a few prescribed functions. The interactions between these module chunks tend to be fundamental to the primary product function and are not explicitly dependent on the technical solution embedded in the module, which allows a more flexible approach to configuring variants of a product. Modular architecture requires well defined interfaces to avoid non-useful combinations of modules. Although our research has its origins in modular architecting, interfacing is the task that is our interest, regardless of the type of architecture.

To develop a modular architecture, three systematic approaches have been identified [6]. Function Structure Heuristics is the first of these methods and uses conversion-transmission function pairs, dominant flow, and branching flows to indicate potential modules and interfaces. This method does not take strategic business decisions into account. The second method is Design Structure Matrix (DSM), which was originally developed by Stewart [7]. DSM maps functions or components against one another through their interactions, as a tool for analyzing given design decompositions [8]. A clustering algorithm is used to numerically minimize interactions between clusters and maximize interactions within a cluster, based on the interfaces between the modules of the product. The DSM method does not consider the needs of the

customer, which is necessary to ensure that the architecture does not fall short in filling market requirements.

Modular Function Deployment (MFD) is the last of the architecture approaches. The MFD approach is composed of five basic steps [9], as shown in Figure 1. Step 1 of the process is to clarify the customer requirements through the use of a Quality Function Deployment (QFD) matrix. The QFD maps the customer requirements against the product properties. Product properties are measureable and controllable entities that allow specification of the product demanded by the customer. Step 2 is to select technical solutions. Technical solutions are the embodiment of the product properties. A function-means tree can be utilized to define the technical solutions, and the results of these decisions are modeled in a Design Property Matrix, first presented in [10]. In Step 3 module concepts are generated. Here the Module Indication Matrix (MIM) considers strategic business module drivers when creating modules. Step 4 is the evaluation of the module concept. In this step an interface matrix is documented, see Figure 2. Finally step 5 improves on the module concept with Design for Manufacturing and Assembly techniques.

Figure 1: Modular Function Deployment Process, adapted from [10], based on [9]

As a systematic approach, MFD, which captures the Voice of the Customer (i.e. using a QFD matrix) and the Voice of the Engineer (i.e. using the DPM), is the method employed to create the modular architectures discussed in this research with a focus on step 4 and the interface matrix.

The term “module” is not new to industry. It is a word that has been used throughout literature for referring to a conceptual unit, “chunk,” or “building block” in a physical or virtual product. However, even though the definition is not consistent from one reference to the next, there is always reference to a combination of the three key conceptual definitions: what the module contains, what the physical limits of the module are, and why the module exists. From Merriam-Webster there are five definitions for a module, where the most relevant definition is as follows: “any in a series of standardized units for use together.” From Baldwin and Clark [1], “a module is a unit whose structural elements are powerfully connected among

themselves and relatively weakly connected to elements in other units.” From [11] a third definition for a module is “an equipment building block with a specific function: execution of one or more basic assembly actions...A module includes its own mounting devices, interfaces etc.” These definitions clearly contain references to modules having interfaces and functions.

The definition of the module to be used for this research takes elements of all of the above definitions into consideration. However, because of the need to capture the VoC in the process of developing an architecture, we have chosen to also include this feature in the definition of a module. Therefore a module is a functional building block with specified interfaces, driven by company-specific reasons [9].

Once the modular architecture has been defined and the modules are set, the interactions of the modules in the concept can then be documented using the interface matrix, see Figure 2. Documentation occurs by making notations about the module interactions. These notations represent a variety of predefined types of interfacing. With this idea, the definition of interface we use in this research is that an interface is a representation of an agreement or contract [1] between conceptual units in the product architecture.

	RC Upper Endcap	RC Lower Endcap	FC Upper Endcap	FC Lower Endcap	RC Skin	FC Skin	RC Inner Panel	FC Inner Panel	Gasket	RC Top Hinge	FC Top Hinge	RC Bottom Hinge	FC Bottom Hinge	Handle	IS&V housing	Water and wiring conduit	Water conduit	Cabinet	Ice bucket mounting plate	Ice bucket	Wire harness
RC Upper Endcap																					
RC Lower Endcap																					
FC Upper Endcap																					
FC Lower Endcap																					
RC Skin																					
FC Skin																					
RC Inner Panel																					
FC Inner Panel																					
Gasket																					
RC Top Hinge																					
FC Top Hinge																					
RC Bottom Hinge																					
FC Bottom Hinge																					
Handle																					
IS&V housing																					
Water and wiring conduit																					
Water conduit																					
Cabinet																					
Ice bucket mounting plate																					
Ice bucket																					
Wire harness																					

Figure 2: Interface Matrix

To standardize on the notations used in an interface matrix interaction taxonomy needs to be established. One interaction taxonomy uses four interface types: spatial, energy, information, and material [12]; spatial is the orientation or adjacency between two modules, energy is any energy transfer that occurs, information is signal or information exchange and material is material exchange between modules.

A second taxonomy is based on five types of module interfacing. The types, from [13], are attachment, positioning, motion, containment, and life-cycle issues. Attachment refers to the physical contact between modules, positioning is the alignment of components, motion is cam-controlled objects,

trajectory of joints, etc., containment is components contained in the same housing, and life cycle is the stages through which a product will endure. The stages are: product planning, designing and developing, implementation, manufacture and assembly, distribution, commissioning, exploitation, and disposal [14].

A third interaction taxonomy has seven types, attachment, transfer, control and communication, power, spatial, and field [12]. Attachment is a physical connection, transfer is a transfer of power or media transfer, control and communication determines how the state of a component is communicated and/or controlled by other components, power is the transfer of electrical power, spatial is the location and volume of a component and field is the ways a component may generate heat, vibration, etc. or other environmental effects that has to be accommodated by other components.

Many of these taxonomies are similar. Each categorizes physical connections, transfers, spatial, and status in slightly different ways, illustrated in Figure 3. For the purpose of this research we propose seven basic types of interfaces. An interface can be defined as attachment, transfer, spatial, command and control, field, environmental, or user. Attachment interfaces, denoted by ‘A’, put the pieces together or connects them physically to one another. An example would be standardized fasteners. Transfer interfaces, ‘T’, provide a conduit for power or media to transfer from one module to the next. Spatial interfaces, ‘S’, determine the boundary between modules. It is the spatial location and volume a component can occupy. Command and control interfaces, ‘C’, are how the state of one module will be communicated to and/or controlled by other components. The functioning of one component can generate heat, magnetic fields, vibrations or other environmental effects, design of other components must account for these effects so a field interface, ‘F’, is used to indicate this. Environmental interfaces, ‘E’, account for product usage affected by heat, cold, and moisture. Stated another way environmental interfaces are the ambient climate environment in which the component must operate and function. Lastly, user interfaces, ‘U’, are the interaction with features within a module system.

The interface types fall into one of three notational levels, types that encapsulate the contents of a module, types that describe the interaction between modules, and a type of interface that describes the human or system needs in engaging the module. Spatial, field, and environmental are types that clarify encapsulation of the module. These define the boundary around the technical solutions and any additional precautions that must be considered. Attachment, transfer, and control and communication are about how two modules interact with each other. A user interface is based on a life cycle approach to the product.

Figure 3: Interface Categories

METRICS

Designers and the processes designers apply in design can make use of metrics to assess the usefulness of results. In this research we are interested in a metric that will guide a designer in determining when an architecture has indirectly reached an appropriate level of granularity. Our intent with this type of metric is to provide a QA tool to determine if too many or too few interface notations have made about a module concept, around a desired index.

An example of a similar type metric is the degree of modularity. Hölttä-Otto and de Weck in [15] use the metric singular value modularity index (SMI) to indicate how modular a product is. If the SMI is closer to 1.0 then information is broadly distributed through the system and is therefore more modular, closer to 0 and the system is more integral.

However, Gershenson, Prasad, and Zhang in [13] argue that the measure of modularity as an abstract number is unimportant until design changes are made. Design changes can have both a positive or negative impact on the architecture. Is there a way to measure this impact?

Net option value (NOV) [1] is such a metric. NOV is the calculation of probable future rewards. Designers can evaluate the impact of a design change on a module using the number of tasks to redesign the module, standard deviation of the potential value, and how visible (number of interfaces) the module is to the rest of the system. While a design may negatively impact a module, it could positively affect the system. The system value is this metric. System value is the sum of the net option values of the modules. The returned value gives the expected return from the customers for the product.

In order for the real customer reaction to be realized the product has to reach the customer. Using a modular

architecture increases parallel product development and decreases lead times. Decreased lead times means faster manufacturing. Ericsson and Erixon in [16] use interface complexity as a measure of modularity. Interface complexity is a normalized value assuming an ideal assembly operation time of 3 seconds. With a lower interface complexity the easier the modules are to put together therefore reducing lead times.

Our approach is to examine the modular composition of a product through the module interfaces recorded for industrially executed modularity projects. These notations are made in the interface matrix discussed earlier, which is generated during a MFD supported project. Our hypothesis is that there are a nominal number of interfaces notations per module, which can be used as a metric.

THEORETICAL FOUNDATION

Our research focus is based on the theory that design is a signing activity [17], our observation being that products are designed by someone and perceived by someone. The hypothesis proposed by this design theory is that there are three levels of notation appearing in a design activity, where each level of notation clarifies a sense of perception the designer experiences about the problem space in which they are solving a problem.

Figure 4: Design Semiosis

At the lowest level of notation, the module should reflect a description of the effect of all the interactions at a module's boundary. At the second level of notation, there should be a description of the external interactions of the module with other modules. Finally, there is a third level of notation that reflects the need for human interaction with the module system. As shown in Figure 5 resulting interface types are sorted into the semiotic notation levels. The idea that we are examining in this research is if it is possible that the three levels represent a nominal number of interfaces per module concept.

When considering the interactions regarding a module, it is enough to say that an interaction exists between two modules. It does not matter how many interface types are contained within that interface or how complex the interface is. Interface

types are an indication of complexity in an interface. The number of interface types per module interface is thought to be three. A module always occupies physical space. This is one of the three theoretical interface types. In order for a module to be part of a modular system it needs to be connected through some means to at least one other module in the system. Therefore it needs, bare minimum, one interface type that enables interaction. And the third type comes from a user's interaction with the module. Hence, the hypothesized theoretical number of three interface types per interaction.

Figure 5: Interface Notation

The total complexity of a module multiplies the theoretical number of interfaces per module by the number of interface types per interface. Given this theory, the complexity metric of interface types to modules is thought to be nine.

For this quality assurance metric to be relevant, it should be applicable to all product architectures. Taking this assumption into account, the area of technical specialty a company has should not influence the outcome of an interface matrix or the calculation of the metrics.

APPROACH

The data collection process illustrated in Figure 6 was created to ensure a systematic approach to gathering information. The data sources referenced for this research are the MFD projects executed since 1996 by Modular Management. The projects are recorded in a variety of different forms: spreadsheet, text document, and presentation slides.

For the purpose of finding a metric, only interface matrices that were generated using the MFD approach were of interest. All interface matrices produced by this method should be similar in nature since the same approach and rigors were used to create them. With this assumption, all interface matrices that were documented using alternative methods were excluded from the analysis.

In order to have a completed interface matrix, the fourth step of the MFD process, Evaluate Module Concept, needs to be completed. Once it was decided that a project had not reached this level of completion, the information was excluded.

Simply because a project had made it through step four of MFD did not guarantee that an interface matrix was done. Since an interface matrix is required for the basis of this research, projects that did not document an interface matrix were also removed from the analysis.

To determine if a company's technical specialty could influence the outcome of an interface matrix, a product classification was added. To standardize this list the Standard Industrial Classification (SIC) was used [18]. This is a body of standards that is governed by Occupational Safety and Health Administration (OSHA), which is a division of the US Department of Labor. An SIC is composed of four levels: division, major group, industry group and classification. The appropriate level of abstraction for the research was determined to be industry group.

As discussed previously the types of interfaces are attachment, transfer, communication, spatial, environmental, field, and user. A few of the matrices used invalid interface types, such as '?'. Conditional formatting highlighted if these interface types existed in the matrix and any invalid types found were removed.

With only valid interface types remaining, the module interfaces could be counted. Using the Excel formula 'countif' any cell within the matrix that contained text was counted as an interface. The count was used to show the number of modules within the system a given module had an interaction. Modules could then be sorted from lowest number of interactions to highest and the total number of module interfaces per modular architecture could be captured. This information was compiled into a master list of project name, year, SIC, number of interfaces, and number of modules.

Lastly the ratio of interfaces per module could be calculated. This ratio is the number of interfaces divided by the number of modules. The resultant was rounded to two decimal places.

In theory one interface can contain anywhere from one interface type to all seven. The complexity of an interface is the number of interface types within that interface. Prior to calculating the complexity, some clean up needed to be done on the matrices. All spaces and commas needed to be removed from the interface matrix. There were some matrices where different module interface types were used. A few examples of this were HY, EL, and Mech referring to hydraulic fluid transfer, electrical transfer, and mechanical transfer respectively. For these alternative types, they needed to be replaced with the standard interface types of attach, transfer, command and control, spatial, environmental, field or user. An attempt was made to be as accurate as possible when replacing the alternative types with a standard type, ensuring that the original intent of the interface description was still intact. The 'len' formula in Excel calculated the length of the text in a cell and returned the number of interface types for a given interface.

Figure 6: Data Collection Process

Because there are seven types of interfaces, there are seven levels of interface complexity. The seven levels of complexity are one, two, three, four, five, six and seven. An interface at complexity level one has one interface type. An interface at complexity level two has two interface types and so on. The ‘countif’ formula was used to compile the number of interfaces at each complexity level.

A module can interface with numerous other modules. Each of these interfaces can have a different level of complexity therefore each module has its own complexity score. This is referred to as the module complexity score (MCS.) To calculate the module complexity score the number of interfaces at each level was multiplied by a complexity constant and summed. The complexity constant is the complexity level. With this rationale an interface at complexity level seven is seven times more complex than an interface at complexity level one. The module complexity score formula is as follows:

$$\text{MCS} = (\# \text{ of } 1\text{'s} * 1) + (\# \text{ of } 2\text{'s} * 2) + (\# \text{ of } 3\text{'s} * 3) + (\# \text{ of } 4\text{'s} * 4) + (\# \text{ of } 5\text{'s} * 5) + (\# \text{ of } 6\text{'s} * 6) + (\# \text{ of } 7\text{'s} * 7)$$

Equation 1: Module Complexity Score

The MCS for all modules was calculated and summed; this is referred to as the total complexity for the system. The average complexity score or average complexity per module was determined by dividing the total complexity by the number of modules.

The compilation of the complexity information enabled a quick QA of the interface totals. Summing all of the interfaces found in the matrix and comparing it to the number documented in the master list verifies that the count was accurate. Any discrepancies were corrected. For the most part,

the errors were due to invalid interface types not being removed from the matrix. Once all projects were verified and confirmed, the complexity score was added to the master list.

RESULTS AND ANALYSIS

To begin the analysis of the data set of forty-four interface matrices identified and collected based on the criteria discussed previously, consider the definition of a product given earlier where a product is physical, discrete, and engineered. Division D of the Standard Industrial Classification (SIC) is Manufacturing. A company that falls within the manufacturing division transforms materials or substances either mechanically or chemically into new products. Or the company assembles component parts of a manufactured product [18]. Therefore a company that falls into this division would produce products that meet the product definition. To verify that the products documented during the MFD process met this definition a plot of the SIC divisions was used, see Figure 7.

It was expected that all of the projects would be for companies within the manufacturing division. However, only 93% of the projects fell there. Looking further at the remaining 7%, it was seen that these were companies whose main business practice lent itself to other SIC divisions but contained a branch dedicated to manufacturing. With this being said, analysis at the Division level confirmed that MFD was done on physical, discrete products.

Figure 7: SIC Division Breakdown

To determine the shape of the frequency distribution, the distribution symmetry, and the modality of the data set a histogram of the calculated interfaces per module ratio needed to be evaluated. Instead of calculating the histogram bins by hand, bins generated by Microsoft Excel were used, Figure 8. The resultant histogram is positively skewed. The data is unimodal, meaning that it is a single data set corresponding to a single distribution.

Figure 8: Excel Generated Histogram

Before continuing with a statistical analysis a check for outliers to needed to be completed. Outliers are defined as any data point or points that appear to be separate from the rest of the data set [19]. For this data set, an outlier would be considered any module architecture that is not at the proper level of abstraction. A rule of thumb to check for outliers is to divide the residual by the estimated error standard deviation σ^2 and any data point with an absolute value greater than 3 is considered an outlier. Figure 9 maps the residual/estimated

error standard deviation ratio and number of modules for each matrix.

There is one data point above the 3 threshold with a ratio of 3.36 and is therefore considered an outlier. Typically one should attempt to transform the data to bring the outlier into line with the rest of the data. Prior to doing this a further review of the data set was conducted. A module is defined as having only one primary function with possibly a secondary or tertiary function. In the outlier data set it was found that the modules had upwards of ten functions per module. The definition of a module was modified and therefore the result was different. The process to generate the outlier was flawed. Due to this the data point was considered an outlier and removed from further analysis.

Figure 9: Outlier Analysis

Now that any outliers have been removed from the analysis the statistical significance of the data should be determined. An F-Statistic would indicate if the number of interfaces is independent of the number of modules. The null hypothesis, h_0 , is the number of modules does not influence the number of interfaces. The alternative hypothesis, h_a , is the number of modules does influence the number of interfaces. SPSS statistical software was used to calculate F. The results are shown in Figure 10. For a confidence interval of 99%, degrees of freedom between groups of 30, and degrees of freedom within groups of 12, $F_{critical}$ is 3.70. $F_{observed}$ is 17.924. Since $F_{observed} \geq F_{critical}$ we reject h_0 . Therefore the number of interfaces is dependent on the number of modules.

ANOVA					
Number of Interfaces	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	490777.9	30	16359.262	17.924	.000
Within Groups	10952.333	12	912.694		
Total	501730.2	42			

Figure 10: F-Statistic

Now having shown that the number of interfaces is dependent on the number of modules, we need to assess if the data is statistically different from each other. To do this a t-test can be conducted. Again with the use of SPSS, a t-test was conducted for both data sets, Figure 11. Interpolating for 42 degrees of freedom and a confidence interval of 99%, t_{critical} is 2.708. For the number of interfaces the t_{observed} is 6.075 and 9.437 for the number of modules. Both data sets have a t_{observed} greater than 2.708. Therefore something more than mere chance coincidence produced these results and the data sets are statistically significant.

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
Number of Modules	43	32.95	22.897	3.492
Number of Interfaces	43	101.26	109.298	16.668

One-Sample Test						
	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
Number of Modules	9.437	42	.000	32.953	25.91	40.00
Number of Interfaces	6.075	42	.000	101.256	67.62	134.89

Figure 11: t-test

To decide what type of regression approach to use, the sort of relationship that occurred between the number of interfaces and the number of modules in a module system needed to be determined. If the data showed a linear relationship then a linear regression could be performed. Exhibition of a nonlinear relationship between the variables would result in the use of a different regression model [19]. The number of interfaces and the number of modules per module system were therefore plotted, Figure 12. Evaluating the plot and considering that as the number of modules increases so does the number of interfaces, a linear regression line seems to be the appropriate model for the data set.

An indication of how well a regression line fits the real data points is the coefficient of determination, R^2 . R^2 will be a value between zero and one. The closer to one the coefficient is the closer the data points are to the regression line i.e. a better fit [19]. The intercept of the regression line was set to zero since if a system has no modules it should have no interfaces. Figure 13 shows the resultant regression line of $y=3.5212x$, with an R^2 value of 0.819. The R^2 value reaffirms the assumption that a linear regression is the appropriate model.

Figure 12: Interfaces vs. Modules

It can then be concluded that the actual ratio of interfaces per module is 3.52. The calculated value is higher than the theoretical estimate of 3. One reason for this was the observed phenomenon where systems that had a greater number of modules also had a greater number of interfaces. In essence, larger modular architectures are more complex (have more interfaces) than smaller architectures. Consequently this complexity in larger systems resulted in a larger interface to module ratio. So the empirical ratio of 3.52 does make sense. Therefore the expected interface to module ratio is 3.50.

Figure 13: Regression Line

Another assumption of this metric is that the ratio of interfaces per module is constant. Meaning it is independent of the number of modules in the system. Plotting the metric to the number of modules, seen in Figure 14, confirms that this assumption is correct.

Figure 14: Interface Ratio vs. Number of Modules

Earlier the Standard Industrial Classification (SIC) was used to confirm that all of the products that generated an interface matrix conformed to the definition of a product. In order to determine if the company's product influenced the resultant interface matrix, the SIC needed to be decomposed further. Again, an SIC is composed of a Division, Major Group, and Industry Group. Revisiting the Division, 93% of the projects fell within the Manufacturing Division, therefore the Division had no influence on the metric and was removed from further discussions.

Moving to the Major Group level of the SIC, it was found that 80% of the data set fell within two major groups, 35: Industrial And Commercial Machinery And Computer Equipment and 36: Electronic And Other Electrical Equipment And Components, Except Computer Equipment. With 80% of the projects composed from these two major groups, the major group has no impact on the metric and was also removed from further analysis.

Lastly, the metric interfaces per module was compared to the Industry Group of the SIC, Figure 15. Refer to Table 1 for the Industry Group key used for Figure 15. From this plot there is no correlation that can be seen between the data in an interface matrix and the area of industry a product hails from. This is consistent with the notion that the architecture should be independent of the area of technical specialty for a company.

Figure 15: Metric by Industry Group

A histogram of the calculated complexity per module ratio shows the shape of the frequency distribution, the distribution symmetry, and the modality of the data set. As before the bins were generated by Microsoft Excel instead of being calculated by hand, Figure 16. The resultant histogram is positively skewed and unimodal.

Figure 16: Complexity Histogram

In this set of data, an outlier would again be any modular architecture not at the appropriate level of granularity based on interface complexity. An outlier check, highlighted in Figure 17, was done using the same rule of thumb as previously stated where the residual is divided by the estimated error standard deviation σ^2 and any data point with an absolute value greater than 3 is considered an outlier. There was one data set that fell above the threshold of 3 with a value of 3.03. Since this violates the rule of thumb the outlier should be removed.

Figure 17: Complexity Outlier Analysis

An attempt should be made to transform the data set to bring an outlier into line with the rest of the data. Prior to making this attempt a further review of the data set was conducted. The guideline for the number of interface types set forth for this research was seven. In the outlier data set it was found that more than seven interface types were used. Therefore the result was different because the definition was modified. A flawed process generated this data set. Due to this the data point was considered an outlier and removed from further analysis.

The statistical significance of the total complexity and number of modules needs to be judged. The F-Statistic null hypothesis, h_0 , is the number of modules does not influence the total complexity of a system. The alternative hypothesis, h_a , is the number of modules does influence the total complexity of the system. Again, SPSS was used to calculate F, Figure 18. For a confidence interval of 99%, degrees of freedom between groups of 30, and degrees of freedom within groups of 12, $F_{critical}$ is 3.70. $F_{observed}$ is 16.531. Since $F_{observed} \geq F_{critical}$ we reject h_0 . Therefore the total complexity of the system is dependent on the number of modules.

ANOVA					
Total Complexity					
	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	4034266	30	134475.520	16.531	.000
Within Groups	97816.667	12	8134.722		
Total	4131882	42			

Figure 18: Complexity F-Statistic

Next a t-test needs to be done on the total complexity and the number of modules to assess if the data is statistically different from each other. SPSS is again used to conduct the t-test, Figure 19. Interpolating for 42 degrees of freedom and a confidence interval of 99%, $t_{critical}$ is 2.708. For total complexity the $t_{observed}$ is 6.489 and 9.504 for the number of modules. Both data sets have a $t_{observed}$ greater than 2.708. So,

there is less than a 1% chance that the results came about by chance.

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
Module	43	32.93	22.720	3.465
Total Complexity	43	310.40	313.653	47.832

One-Sample Test					
	Test Value = 0				95% Confidence Interval of the Difference
	t	df	Sig. (2-tailed)	Mean Difference	Lower Upper
Module	9.504	42	.000	32.930	25.94 39.92
Total Complexity	6.489	42	.000	310.395	213.87 406.92

Figure 19: Complexity t-test

To determine the relationship between the complexity of a system and the number of modules, a scatter plot was created, Figure 20. Considering that it is reasonable to assume that as the number of modules increases so should the total complexity of the system, the plot supports that a linear regression line could be an appropriate model.

To ascertain the regression line, the intercept was set to zero because if the system has no modules it should have no complexity. The resultant regression line is $y=9.7561X$ with an R^2 value of 0.7813. The high R^2 value confirms that the linear regression is appropriate.

Figure 20: Total Complexity vs. Number of Modules

It can be concluded that the actual ratio of complexity to modules is 9.75. The original assumption for the complexity per module was 9. This was based on the idea that a module should have three module interfaces each having three types. The research showed that the expected number of interfaces per module was 3.50. Therefore, the expected complexity per module should be 10.50. The difference between the adjusted theoretical value and the actual is 0.75. Since not every module

is going to have a user interface, the expected ratio 9.75 makes sense.

Figure 21: Total Complexity Regression Line

Another assumption of the complexity ratio, similar to that of the interface ratio, is that it should be constant or independent of the number of modules in a system. Plotting the ratio against the number of modules, Figure 22, affirms this assumption.

Figure 22: Complexity Ratio vs. Number of Modules

While the SIC did not have an impact on the interface ratio, it needed to be verified that this was also true for the complexity ratio. From the previous analysis, the Division and Major Group have no influence on the ratio since a majority of the projects are from similar areas. The Industry Group is the level of interest here. Refer to Table 1 for the Industry Group Key. After plotting the total complexity against the Industry Group, Figure 23, it became evident that there is no correlation between the product and the complexity.

Figure 23: Total Complexity by Industry Group

Industry Group	
1	335: Rolling, Drawing, & Extruding of Nonferrous
2	349: Miscellaneous Fabricated Metal Products
3	351: Engines And Turbines
4	353: Construction, Mining, And Materials Handling
5	354: Metalworking Machinery And Equipment
6	356: General Industrial Machinery And Equipment
7	358: Refrigeration and Service Industry Machinery
8	359: Miscellaneous Industrial And Commercial
9	363: Household Appliances
10	371: Motor Vehicles And Motor Vehicle Equipment
11	374: Railroad Equipment
12	382: Laboratory Apparatus And Analytical, Optical, Measuring, and Controlling Instruments
13	504: Professional And Commercial Equipment And Supplies
14	651: Real Estate Operators (except Developers) And Lessors

Table 1: Industry Group Key

SUMMARY AND CONCLUSION

The metrics of interfaces per module and average complexity per module for projects building a modular architecture using the Modular Function Deployment method has been found to be 3.50 and 9.75 respectively. These metrics allow a designer to discover if the architecture is at the right level of granularity as well as if the number of interface specifications is manageable. However, these metrics have not been tested using the two alternative methods of modular architecting, Function Structure Heuristics and Design Structure Matrix (DSM).

Function Structure Heuristics does not generate an interface matrix. It could be possible to use the function structure diagram to collect module and interface information.

Modules would correspond to the functions and the interfaces would correspond to the individual flows. From here the ratio of interfaces per module could be calculated.

DSM does generate an interface matrix. This matrix includes both feed forward and feedback information. To calculate the interface to module ratio the feedback would need to be removed. Once these interactions are removed the remaining interactions would be the interfaces. The modules would refer to the tasks listed in the DSM. Calculation of the ratio could be done with this information.

Further investigation is needed to determine if these methods would produce a similar metric. This is a topic for future research.

Neither Function Structure Heuristics nor DSM documents an interface's complexity. The metric of average complexity per module is exclusive to MFD generated modular architectures. Another area of future research would be to find if there is a way to calculate interface complexity using these methods.

Interfaces are the key to a successful product architecture. They must be robust, stable over time and standardized. To ensure this, interface specifications need to be written. The specification is a contract between design teams that have negotiated the terms of the interface. Interface specifications are also design tasks that need to be planned for. To write fifteen to forty interface specifications per module would become an overwhelming task. Not to mention that as design cycles and lead times become shorter this becomes unrealistic. Armed with the expected number of interfaces per module and the average interface complexity per module the scope of a project is known.

The metrics determined here aid in the verification of a product architecture. Too many interfaces or too few and the project team could have issues as the project progresses. Sorting modules by the number of interfaces per module will show if there are modules in the matrix that do not have any interfaces. This is also an indication that an interaction may have been missed, preventing negative impact on the design cycle later on. It could also highlight that the technical solution can be embodied within another module.

Lastly, the complexity would indicate if there are modules that need further analysis to simplify and document. High complexity could indicate that the module contains too many functions and needs to be broken down further. No complexity could indicate that a module may need to become part of another module or that it is not part of the modular architecture.

All of these items strengthen the interface portion of an MFD project and help to build a solid foundation heading into the specification writing phase. With a solid interface matrix a DSM can be run effectively and a design project can be planned with greater confidence.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Gunnar Erixon, Fredrik Börjesson, Dr. John Niethammer, Anders Leine, and Ola

Nordstrom (Modular Management USA) as well as Brandon Imsdahl and Josh Durand for their feedback and guidance in the writing of this paper.

REFERENCES

- [1] Baldwin, C.Y., and Clark, K.B., 2000, *Design Rules: Volume 1. The Power of Modularity*, Cambridge, MA, MIT Press.
- [2] Ulrich, K. T., and Eppinger, S. D., 2003, *Product Design and Development*, McGraw-Hill/Irwin, Singapore.
- [3] Baldwin, C.Y., and Clark, K.B., 1997, "Managing in an Age of Modularity," *Harvard Business Review*, 75(5), 84-93.
- [4] Sundgren, N., 1999, "Introducing Interface Management in New Product Family Development," *Journal of Product Innovation Management*, 16(1), 40-51.
- [5] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2007, "Are Your Engineers Talking to One Another When They Should?" *Harvard Business Review*, 85(11), pp. 133-142.
- [6] Hölttä-Otto, K., 2005, "Modular Product Platform Design," Helsinki University of Technology, Department of Mechanical Engineering, Machine Design, Espoo.
- [7] Stewart, D.V., 1981, *Systema Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, New York.
- [8] Pimmler, T. U., and Eppinger S. D., 1994, "Integration Analysis of Product Decompositions," *ASME Design Theory and Methodology Conference*, Minneapolis, MN.
- [9] Erixson, G., 1998, "Modular Function Deployment - A Method for Product Modularisation," The Royal Institute of Technology, Department of Manufacturing Systems, Assembly Systems Division, Stockholm.
- [10] Nilsson, P., 1998, "The Chart of Modular Function Deployment", 4th Workshop on Product Structuring, Delft University of Technology, 22-23 October.
- [11] Vos, J.A.W.M., 2001, *Module and System Design in Flexibility Automated Assembly*, DUP Science, Netherlands, pp. 23, Chap. 2.
- [12] Bettig, B. and Gershenson, J. K., 2006, "Module Interface Representaion," *ASME Design Engineering Technical Conferences*, Philadelphia, PA, ASME, DETC2006-99554.
- [13] Gershenson, J. K., Prasad, G. J., and Zhang, Y., 2004, "Product Modularity: Measures and Design Methods," *Journal of Engineering Design*, 15(1), pp. 33-51.
- [14] Hubka, V. and Eder, W.E., 1996, *Design Science*, Springer, London.
- [15] Hölttä-Otto, K. and de Weck, O., 2007, "Metrics for Assessing Coupling Density and Modularity in Complex Products and Systems," *ASME Design Engineering Technical Conference - Design Theory & Methodology Conference*, Las Vegas, NV, ASME, Paper No. DETC2007-34871.

- [16] Ericsson, A. and Erixon, G., 1999, *Controlling Design Variants: Modular Product Platforms*, Society of Manufacturing Engineers, Dearborn.
- [17] Lange, M. W., 2001, "Design Semiosis Synthesis of Products in the Design Activity," Department of Machine Design, Royal Institute of Technology, Stockholm.
- [18] Occupational Safety and Health Administration, US Department of Labor, "SIC Division Structure," US Department of Labor Occupational Safety and Health Administration Website. [Online] February 12, 2009. [Cited: February 11, 2009.] http://www.osha.gov/pls/imis/sic_manual.html.
- [19] Hayter, A. J., 2002, *Probability and Statistics for Engineers and Scientists*, Duxbury, Pacific Grove.