ASME 2016 International Design Engineering Technical Conferences and Design Automation Conference
IDETC2016
August 21-24, 2016, Charlotte, North Carolina

IDETC2016-59808

INCORPORATING CONSTRAINTS IN SYSTEM MODULARIZATION BY INTERACTIVE CLUSTERING OF DESIGN STRUCTURE MATRICES

Roozbeh Sanaei
Singapore University of Technology and Design
Singapore
Roozbeh_Sanaei@mymail.sutd.edu.sg

Kevin Otto
Aalto University
Finland
Kevin.Otto@aalto.fi

Katja Hölttä-Otto
Singapore University of Technology and Design
Finland
Katja.Holtta-Otto@aalto.fi

Kristin L. Wood
Singapore University of Technology and Design
Singapore
kristinwood@sutd.edu.sg

ABSTRACT

Modularity is an approach to manage the design of complex systems by partitioning and assigning elements of a concept to simpler subsystems according to a planned architecture. Functional-flow heuristics suggest possible modules that have been demonstrated in past products and systems~~and~~, but using them still leaves it to the designer to choose which heuristics make sense in a~~n~~ certain architecture. ~~This constitutes an opportunity for a designer to take other constraints and objectives into account.~~ With large complex systems, the number of alternative groupings of elements into modular chunks becomes exponentially large and some form of automation would be beneficial to accomplish this task. Clustering algorithms, using the design structure matrix (DSM) representation, search the space of alternative relative positioning of elements, and present one ideal outcome ordering which "optimizes" a modularity metric. ~~Beyond the problems of lack of interactive exploration around the optimized result, s~~Such approaches also partition the elements in an unconstrained manner. Yet, typical complex products are subject to constraints~~ ,which invalidate the unconstrained optimization~~. ~~Such~~ such as ~~rchitectural partitioning constraints include those associated with external force fields including~~ electric, magnetic, or pressure fields, and ~~that constrain some functions to perform or not perform in different regions of the field~~. ~~There are also~~ supplier constraints where some components cannot be easily provided with others. ~~Overall, it is difficult to simply embed all objectives of modular thinking into one metric to optimize.~~ We develop a new type of interactive clustering algorithm approach considering multiple objectives and partitioning constraints. Partitioning options are offered to a designer interactively as a sequence of clustering choices between elements in the architecture. A designer can incorporate constraints that determine the compatibility or incompatibility of elements by choosing among alternative groupings progressively. Our aim is to combine computational capability of clustering algorithms with the flexibility of manual approaches. Through applying these algorithms to a MRI machine injector, we demonstrate the benefits of interactive cooperation between a designer and modularity algorithms, where constraints can be naturally considered.

Formatted: Indent: First line: 0"

NOMENCLATURE

Design Structure Matrix (DSM)	Matrix of Interactions between pairs of Components/Functions
Cluster (noun)	Collection of elements, equivalent to module.
Cluster (verb)	Generate a set of clusters by means of an algorithm
IGTA	Idicula-Gutierrez-Thebeau Algorithm for clustering Component-DSM
IGTA+	Modification of IGTA that includes two algorithmic changes, SMA and ITC
Intra-cluster Cost	Overall cost of Interactions between Elements that belong to the same cluster
Extra-cluster Cost	Overall cost of Interactions between Elements that belong to different Clusters

INTRODUCTION

~~—~~Modular product architectures offer various benefits. For example, modular platform architecture enables enterprises to respond effectively to variety in market needs with minimum increase in development effort (i.e. time and cost) [1–4]. Product modules can also be updated ~~time by~~over time, replaced as they wear out or are swapped to add functionality [5]. Furthermore, design process~~es~~ can be simplified by decoupling the sub-processes of designing different modules to be conducted by separate groups of designers in parallel. Design cycle time can be also reduced considerably by utilizing modules designed earlier as it allows different parallel design processes simultaneously and autonomously next to each other. Modularity in product architecture may also lead into modularity in organization design [6].

~~Given these benefits, t~~The modular design process typically begins in the conceptual design phase. Modules are often defined ~~once the~~iteratively as product concept and ~~the~~ main components and interactions become clear. A system architecting task then consists of clustering the multitude of product elements into clustered modules such that the defined modules are effective for future generations of the product, development of new products, and detail design and maintenance purposes [7].

Interactions between components can be captured by a graph where the nodes represent elements of a design (i.e. components or functions)~~, and~~ where its edges denote interactions among these elements [8–10]. Such interactions can be physical connections, spatial relationships, or flow of energy, information or material from one element into another. For example, schematics depict components and their interconnecting flows, or function structures depict product functions and their interconnecting flows. Such graphs can be displayed visually and modularity defined manually by grouping their elements manually using heuristics [9,10]. ~~Typically in such a manual approach to partitioning elements into modules, architectural constraints are considered by the designer, such difficult or easily allowed bundling of components. This reasoning can be incorporated directly through smart choice of partitioning.~~

For large complex system, the number of elements and their interactions increases exponentially, ~~and therefore~~where manual heuristics leave so many options that finding an optimal choice is no longer trivial. ~~Furthermore, going through~~Exploring the large number of options ~~might be~~grows increasingly infeasible without having some form of computerized automation. A sorting metric can be useful to prioritize one modularity partitioning option over another. Clustering algorithms enumerate large numbers of architectures and search for one that optimizes a certain modularity metric. These algorithms take as input ~~the~~ an adjacency matrix of the graph representing the system, the Design Structure Matrix (DSM). The convenience of DSMs for representing large systems and the possibility of their manipulation using commercially-available software makes DSMs ~~the representation of choice~~a preferred representaton for complex system modularity analysis. DSMs have been used for decomposition and integration purposes in design of gas turbines [11], printing systems [12,13], helicopters[14] and jet engines [15].

When trade-offs between different objectives are of concern, multi-objective optimization might be necessary [16]. Modularity metrics in this context represent the objective of encapsulating interactions within modules and enhancing independence among modules. Various metrics of modularity combine these two factors in different forms [17]. Regardless of the choice of the modularity metric, however, typical complex products are subject to other constraints often critical for consideration. For example, some components cannot be placed in the same module with others due to pressure or temperature fields [18]. There can also be supplier limitations in bundling some components with others. These constraints can constitute a significant driving factor during product architecture decisions [16].

Comment [KLW1]: I don't understand this sentence and the point you are making.

Since such architectural constraints are often in different forms for different designs, incorporating the wide variety of such constraint ~~forms~~ into a search algorithm metric might not be practical. On the other hand modularizing a complex system manually is also impractical to consider the multitude of alternatives. We therefore seek to allow designers to participate with an interactive modularization algorithm ~~in some way~~ that can allow for interactive management of different architectural constraints and objectives. In this paper we suggest a set of mechanisms by which a designer can actively supervise and direct the modularization as a clustering algorithm computation proceeds, to impose constraints that could not be otherwise taken into account in conventional clustering algorithms.

The ~~rest~~ remainder of the paper is organized as follows. First, we present an overview of different modularization approaches and modularity metrics. We then describe our proposed control mechanisms which enable a designer to impose constraints on the product architecture. Finally we demonstrate implementation of these controlled algorithms on MRI machine injector as an example of complex product architecture.

RELATED WORK

We consider research in modularity in two categories of manual methods using architectural schematics or algorithms operating on matrices representing the schematic graph (design structure matrices). Manual approaches start~~s~~ with the architectural schematic, and then consider possible alternative modules that can be defined by grouping elements according to a set of clustering rules based on the graph structure of the flows between elements [9,10,19,20] . However, the rules developed provide alternative suggestions and are not unique axioms for grouping elements. The *dominant flow* and *conv~~v~~ert-transmit* rules, for example, are maximal, they define upper limit boundaries that modules should not exceed. A module inside these boundaries is consistent with these rules. With the dominant flow rule, a serial chain of functions defines a maximal module, but also any sub-set of this chain would still

be consistent with the heuristic. The functional modularity approach thereby naturally provides room for involving designer insight and judgment. They can choose how maximal to make a module along the dominant flow path and take other considerations and constraints into account.

The concept of manual modularity analysis was first introduced by Stone *et al.* [7] using function structures [21]. They suggested finding the dominant flow, branching flows, or conversion-transmission function pairs as three heuristics for identification of modules in a function structure schematic. Later, three additional heuristics were purposed by Zamirowski and Otto for finding common modules in a product family [10] namely, finding repetitive functions within a single product, common functions across products, and unique functions that are found only in one product. Linked function pairs ~~is~~ are ~~also~~ suggested ~~*et al.*~~ as a module by McAdams *et al.* [22]. Sudjianto and Otto also introduced brand signature rules for bundling elements into modules that carry the brand identity across multiple products [19].

Given an architectural schematic, the total number of alternative ways the elements can be clustered into modules is equal to the Bell number of the graph which is known to grow faster than exponential with the number of elements. Although manual partition rules limit the number of possibilities to a fraction of this number, often this still leave too many options to consider all. Without some form of intervention judgement for eliminating alternatives, an optimal choice is not possible for complex systems with large number of elements.

To address means of using interactive clustering algorithms, we explore those that cluster vertices in a graph. This graph clustering problem is formulated in various ways as network algorithms found under different names, such as "Graph Clustering" [23], "Graph Partitioning" [24], or "Community Detection" [25].

Graph clustering algorithms can be categorized into two classes, where the first define a clustering metric and implement a search algorithm to optimize its value. This problem is NP-Hard [26], and so heuristic algorithms are used to find a sufficiently good solution. Different heuristics are used depending on the application [27]. Newman and Girvan present a graph-theoretic metric of modularity [28]. Information theoretic measures of modularity have been used by Yu et al. [29] and Wang and Antonsson [30]. Search methods used include extremal optimization [31], genetic algorithms [32], mathematical programming [33], simulated annealing [34], spectral methods [35] and tabu search [36]. A comprehensive literature review of graph partitioning was done by Fortunato [25].

The second class of clustering algorithms do not explicitly define a clustering metric but instead deploy an iterative procedure for progressively constructing modules. Partitions are computed and accepted when satisfactory to a human judge, or considered acceptable according to a set of benchmarks. These algorithms include hierarchical clustering algorithms, either bottom-up agglomerative algorithms in which clusters are merged when they have strong interactions or top down divisive algorithms in which graphs cut iteratively into two partitions with low interaction [25].

In product architecture literature, works also consider the same basic problem of clustering a graph. Here, there are two main types of modularity metrics defined, where the first ~~which~~ considers interactions among elements and the degree of independence of a module from the rest of the system. The second type considers the similarity between modules so that elements with similar properties are clustered together. The intended benefit is ~~t~~the reusability of a module or the possibility of using the same materials, manufacturing processes, or suppliers. For example, a metric by Mattson and Magleby [37] minimized the type of interfaces. Aarnio [38] developed an algorithm to consider functional, economic and product variety issues. Hollta-Otto and Salonen considered both independence and similarity of components within modules and considered the internal intra-modular connectivity [39].

Overall, while different modularity methods have been developed, it is difficult for a single metric to incorporate all significant concerns of the different modularity concerns, independent of additional external criteria. For example, having internally integrated modules and keeping the independence of each module from the rest of the system are usually conflicting factors. One approach is to establish a trade-off between the different modularity objectives as a multi-objective optimization formulation to find non-dominant solutions [16]. However, it is practically challenging to have several objective variables in a same optimization.

On the top of the different objectives that define different ideal modularity clustering, ~~the~~ products can be subject to other technical constraints in terms of mass, volumet~~ric~~ and power efficiency of individual modules in the architecture. ~~Also, t~~There ~~can be~~are likewise constraints associated with ~~force fields like~~ electric, magnetic, or pressure fields that limit~~s~~ functionality~~ies~~ ~~performing~~ in different portions of the architecture. There can also be constraints in bundling different components together due to limitations of suppliers [18]. To address ~~this~~these issues, we next develop an approach for interactive cooperation between a designer and a modularity algorithm that analyses the remaining un-partitioned portions of the architecture and indicates which interfaces are most and least suitable for partitioning. This enables a designer to incorporate constraints which could not be straightforwardly coded into conventional clustering algorithms.

METHODOLOGY

With interactive algorithms, there can be multiple approaches to interacting with the computation to pursue a optimum. Designers may prefer alternative approaches, and switch among them while exploring the search space. We offer three alternative interactive clustering algorithms. The first allows a designer to make partitioning decisions, and evaluates the result to highlights the impact of partition a~~n~~ remaining link in the architecture. The second refines this by suggesting which candidate is best to partition and the result of doing so. The third reverses this and uses an algorithm to partition the

Comment [KLW2]: Need to include also the journal versions: Stone, R. B., Wood, K. L., and Crawford, R. H., "A Heuristic Method to Identify Modules from a Functional Description of a Product," Journal of *Design Studies*, Vol. 21, No. 1, pp. 5-31, January 2000;
and

Stone, R. B., Wood, K. L., and Crawford, R. H., "Using Quantitative Functional Models to Develop Product Architectures," Journal of *Design Studies*, Vol. 21, No. 3, 2000, pp. 239-260.

Comment [KLW3]: Good discussion in this secton, but why don't you introduce the algorithms formally?

architecture, and in presenting the result, allows the designer to negate partitions made.

First Approach: Algorithmic Evaluation with Designer Proposals

The first approach is based on the notion that in each clustering scheme, the module boundaries cross certain component connections, and these will define the module interface. In this approach, a designer selects the connections to be cut by module interfaces. An algorithm helps the designer by associating to each connection a value indicating how strongly it should also be cut to further split a module into smaller modules. Every time designer cuts a connection, the values of the remaining links are updated. A useful recommendation value is the rank indices of likelihood that a connection forms a module boundary in a "good" architecture. A "Good" architecture can be defined by two criteria: it is one in which its modularity metrics are within a pre-defined acceptable range, and it is non-dominated by other clustering schemes and so on the Pareto-frontier of metrics of interest.

For example, the IGTA+ algorithm is a common approach to cluster~~ing~~ DSMs [16]. IGTA+ has two clustering metrics, intra-cluster and extra-cluster cost, typically weighted into an overall metric. The extra-cluster cost indicates the strength of connections among modules. The intra-cluster cost measures the integrity of modules and is calculated as weighted summation of number of interactions within individual modules.

In this first approach, the designer constrains any two elements to be in separated or similar modules. The separation constraints are imposed through modifying *ClusterBid* in the IGTA+ algorithm. *ClusterBid* determines how fit a selected component is with each cluster. We modify *ClusterBid* such that it produces of zero when there is a separation constrain between the element of choice and any element of chosen cluster.

Similarly, constraints imposing components to be in the same module can be simply implemented by merging the two elements into one. An alternative approach is to assign to the component connection a very-large weight in the DSM, but this can lead to ill-posed convergence problems when running IGTA+ (and also other clustering algorithms). To compute recommendation values, we associate a high recommendation with a high number of good architectures possible with that cut. We compute this by follow the standard approach and execute IGTA+ algorithm for a large number of runs. We the calculate the recommendation value as the number of "good" architecture that follows the imposed constraints divided by the total number of "good" architectures that arose in the course of running IGTA+.

Figure 1 illustrates the concept. As ~~visible~~ shown in the figure, initially the algorithm suggests connections e-f, e-d, a-b as the best connections ~~to be cut by~~to introduce an interface. According to ranking, f-h and g-j can be alternatively cut by boundaries of modules with equal priorities. When the designer potentially ~~thinks differently~~considers other choices of interfaces, and selects the connection c-d to be cut, the algorithm then updates its ranking to e-f, a-b and a-d. By ~~putting~~ introducing another constraint on b and d to be on separate modules, the ranking changes again to e-f, b-e and a-d. Finally when the designer enforces that g and j are to be in the same module, the algorithm modifies its ranking to e-f, e-b, f-h, a-d and h-j. After ~~cutting these connections~~introducing interfaces by the designer, the resulting modularity ~~results in~~includes (a, b, c) , (d,e), (f,g,j) and (h,i) as modules.

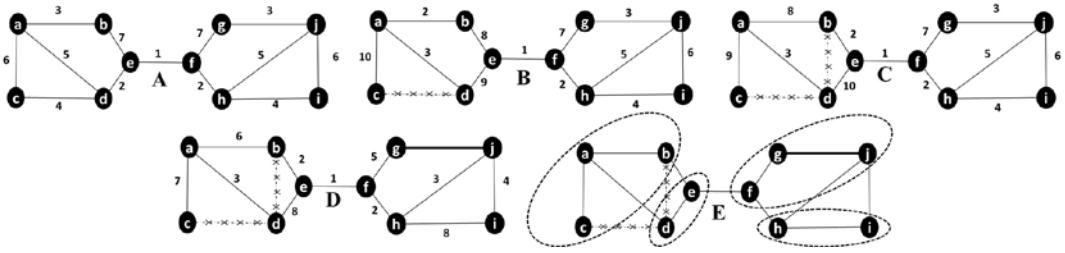

Figure 1. Modularization in the first approach

Second Approach: Hierarchical Partitioning based on Designer Decisions

The second approach follows a hierarchical theme. However, as shown earlier in [16]. A hierarchical approach might not lead to an optimal clustering. When an architecture is partitioned into several modules, splitting one into two modules is not necessarily the best clustering; the previous clusters may need to be redone. This in turn implies partitioning decisions cannot be done in a greedy way for every division, and we need to estimate end-results. Nonetheless, we review this approach given the popularity of K-means hierarchical clustering and similar approaches.

Here, the algorithm ranks alternative bi-sectioning of the product graph. These ranks are based on the likelihood that a particular hierarchy results into a "good" architecture. Calculation of these rank values is performed as in the first

Comment [KLW4]: Reword. Awkward sentence.

Comment [KLW5]: Have you sufficiently introduced this term?

approach by counting the number of good architectures after imposing the hierarchy. Figure 2 explains the second approach graphically. Initially the algorithm suggests two alternative architectures, the first option of ~~putting~~ placing (a, b, c, d, e) in one module and (f, g, h, ~~i~~I , j) in another module. A second option is to ~~put~~ place (a, c) in one module, ~~and~~ (b, e, f, g) in a second module and (h, i, j) in a third module. The first option is consistent with all constraints ~~so designer may select it~~. Then the algorithm moves on to suggest two new options for each section of the graph. However, ~~none of them~~neither of these ptions are consistent with the constraints, so now the designer imposes constraints similar to those in the first approach. The algorithm updates recommendations based on the constraints imposed for each section of the graph. One of these architectures is then selected.

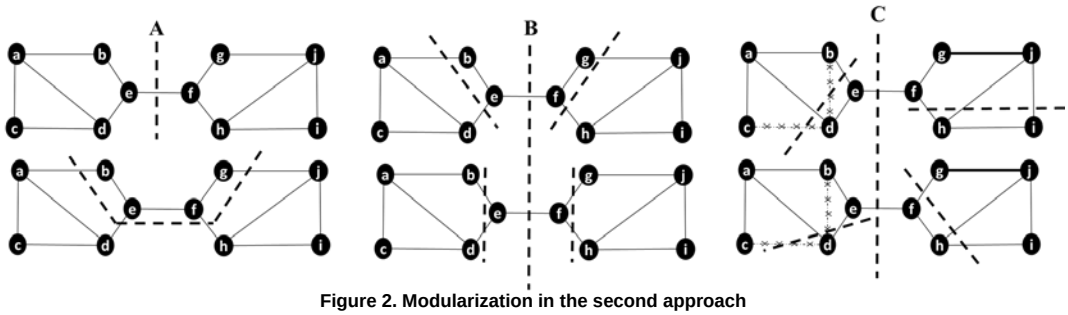

Figure 2. Modularization in the second approach

Third Approach: Algorithmic Proposals with Designer Evaluation

In the third approach, the algorithm first presents modularizations which are close to optimal according to the partitioning algorithm weightings but without taking any constraints into account. This is done simply applying the IGTA+ algorithm to find the architecture with maximum weighted sum values of the modularity metrics among the large number of generated architectures. Then the designer can choose one architecture and set the constraints considered violated. Then the algorithm can be re-applied and it will suggest a new optimal clustering given the imposed constraints. This approach establishes an interactive process between the algorithm and the designer which can repeat until reaching to the desirable modularization.

The third approach is depicted in Figure 3 for the same example as before. Initially the algorithm suggests three different alternative architectures to the designer with respectively two, three and four modules. None of these architectures are consistent with given constraints so designer selects the architecture with four modules, and enforces that d cannot be placed in the same module either with b or c. Afterwards, the algorithm ~~put~~ places (e,d) in one module and (a, b, c) in a separate module to follow given constraints. The designer then enforces one more constraints on the architecture so that g and h should be always in the same module. The algorithm then updates its solution which is optimal given the constraints imposed.

As observed in this example, different purposed approaches can reach the same optimal architecture given a particular set of constraints in different ways. There are always alternative approaches to interactive optimization and trade-study analyses. We demonstrate three alternatives here. Next, we consider an industrial case study of a medical contrast injector.

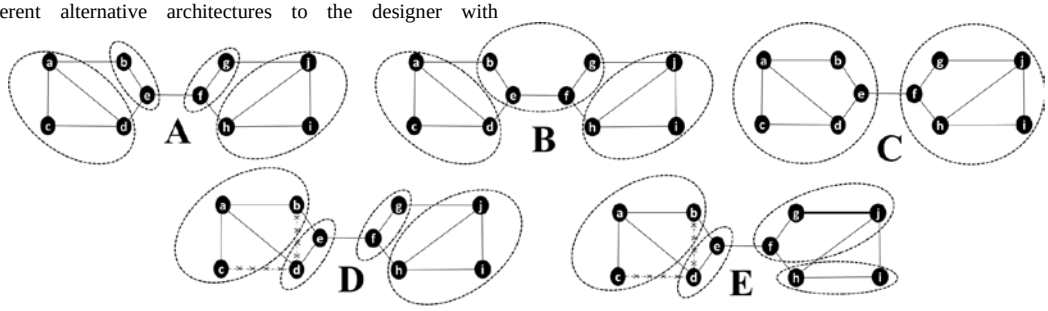

Figure 3. Mechanism of product modularization in the third approach

CASE STYDY: MRI INJECTOR

~~—~~Here we consider an MRI injector used in magnetic resonance imaging as a ~~more~~ complex product or system to demonstrate how different constraints can be incorporated into product modularization through the proposed approach~~es purposed~~. ~~The~~ A function structure schematic of the MRI injector is shown in Figure 4. The system is controlled via a screen, keypads and speakers on the left side of the structure. The power supply including the power cord and rechargeable battery is located in the top left corner of structure. The injector head is in the middle of the structure and is connected with material flow chains of contrast and saline flush for patient injection as placed in the right-side of structure.

The MRI injector is subject to various architectural constraints as posed by different fields inherent to its operational setting. The first constraint is imposed by the magnetic imaging field. The magnetic field strength is typically around 3 Teslas in clinical applications and might be even higher when the system is used for research purposes. The presence of such magnetic field limits materials that can be used in the area of MRI machine to plastics, ceramics, and some non-magnetic metals. Another constraint is patient sterility; materials that are injected to patient body must be sterile and cannot be placed together with non-sterile components.

The MRI injector design structure matrix representing the function structure of Figure 4 is a symmetric matrix of 50 rows and columns representing the 50 subfucntions in Figure 4, and 110 non-zero entries representing the interconnecting flows of ~~figure~~ Figure 4, where there are 30 flow inter-connections among elements and also 50 diagonal elements equal to 1).

For this problem, repeating the IGTA+ algorithm 10^5 times executed in 35 minutes on a computer with two Intel(r) Xeon(r) (e5-2620 v3 @ 2.40ghz) CPUs, each with six physical cores (2 logical cores per physical) with 64 GB RAM using 20 threads running in parallel using MATLAB R2014b. The IGTA+ parameters [16] were set according to Table 1. Based on the Pareto-frontier, we consider any architecture with intra-cluster cost lower than 1700 a.u. and extra-cluster cost less than 2400 a.u. as a potential "good" architecture.

Parameter	Value
pow_cc	1
pow_bid	-2
max_cluster_size	DSM size
rand_accept	2×DSM size
rand_bid	2×DSM size
stable_limit	2
max_repeat	1000

Table 1. List of parameters and values used to run IGTA+ algorithm

Applying the first approach, the initial suggestion of algorithm is ~~to cut the connection~~to introduce an interface between control main units and control GUI, control main unit and filter signal with both the couple shaft and transmit torque connections. Typically, the low magnetic field zone is about 1 meter away from high magnetic field zone and so moving the couple shaft is not feasible considering the torque transmission through such an elongated shaft and shaft deflection. This deflection would adversely affect~~s~~ accuracy of flush and contrast dosage which is a critical factor to the injector performance. For this reason, we enforce a constraint that the connection between couple-shaft and transmit torque should not be crossed by a module interface. Adding this constraint changes the recommendations to now insert an module interface between convert electric energy to torque and couple shaft, and a second between connection card and signal volume status. Notice here the convert-transmit rule from the function based research is violated; the chain of three functions convert e to torque, couple shaft, and transmit torque is not recommended as a module as it produces low intra cluster cost and high extra cluster cost. Given the field constraints, it is easier to break it into submodules.

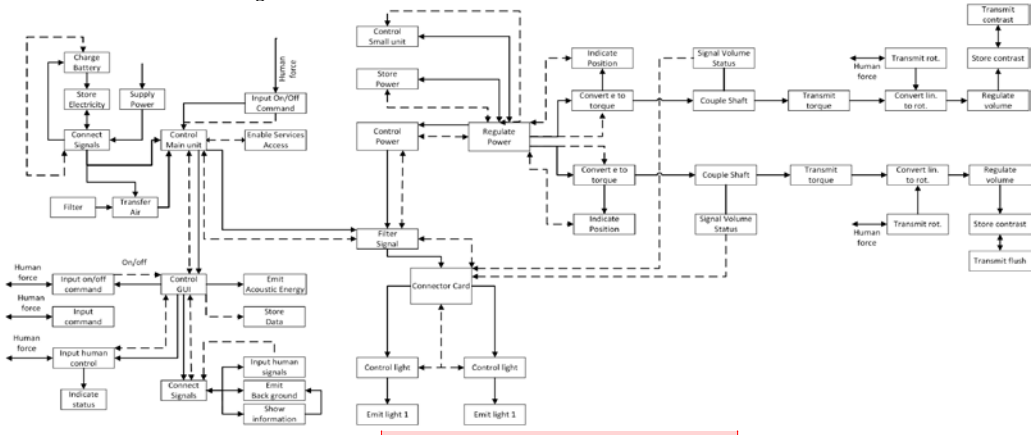

Figure 4. MRI Machine Injector Function Structure

Comment [KLW7]: Never defined. Need to provide the mathematical formulation.

Comment [KLW8]: It is a heuristic, not a rule. It suggests modules and interfaces.

Comment [KLW9]: ????????

Comment [KLW6]: This is not a well structured sentence. Reword.

Comment [KLW10]: Why do you have double arrowed "Human Force." Human force can enter the system, but the system cannot impose a human force. The system is not human.

Moving onward, ~~as sterile functions,~~ the *transmit flush* and *transmit contrast* must be in sterile field. This can be simply enforced by indicating the *transmit contrast* should not be placed in the same module with *transmit torque* (or any mechanical function). Once this constraint is imposed, the algorithm suggests to insert a module boundary between the *store contrast* and *regulate volume* functions. Generally, having two functions in one module is preferred over having individual functions as modules to reduce total intra-cluster cost. The next module boundaries suggested by the algorithm are between *regulate power* and *control power* and between *connect signals* and *control GUI.* Neither are subject to any particular constrain and so are acceptable. The final modularity scheme is shown in Figure 5, as developed using the first approach.

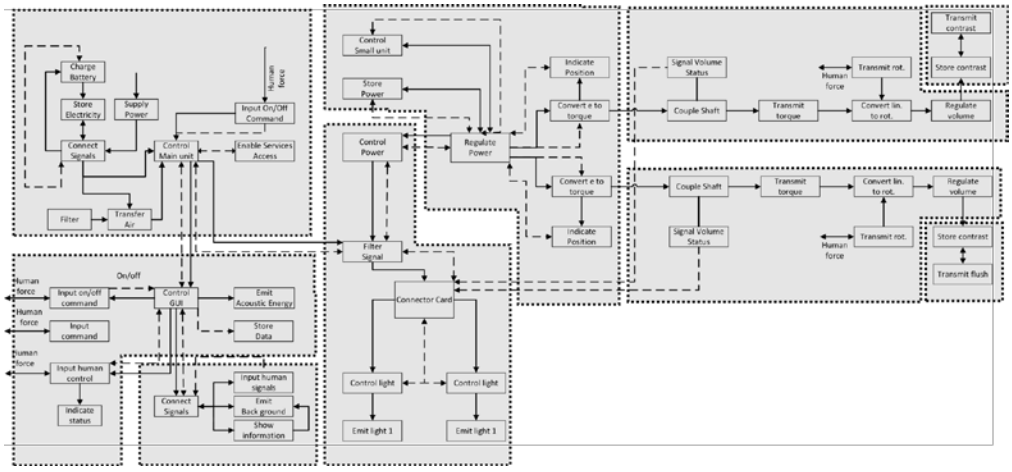

Figure 5 MRI Machine Injector Function Structure with modules created by first approach

Alternatively, a designer may prefer to approach the problem by a sequence of partitioning the schematic, the second approach. Here, alternative bi-sections are offered by the algorithm as shown in Figure 6.1. The bi-sections selected are shown in the second graph in Figure 6.2. The magnetic field constraints can be taken into account simply by the right choice of bisections, without explicitly forcing constraints into the algorithm. However, the sterility constraints must be explicitly set, otherwise the bisections would violate the sterility requirements. By imposing the sterility related constraints, module boundaries between *G-3* and *H-3* are suggested.

Notice in the first approach, the method ranks all links and reports them to the designer in a sorted list so multiple links can be cut simultaneously. In the second approach, the algorithm only proposes various ways to bi-section the structure and only one would be chosen for the sectioning to proceed in a hierarchical manner.

Comment [KLW11]: Where is the sorted list?

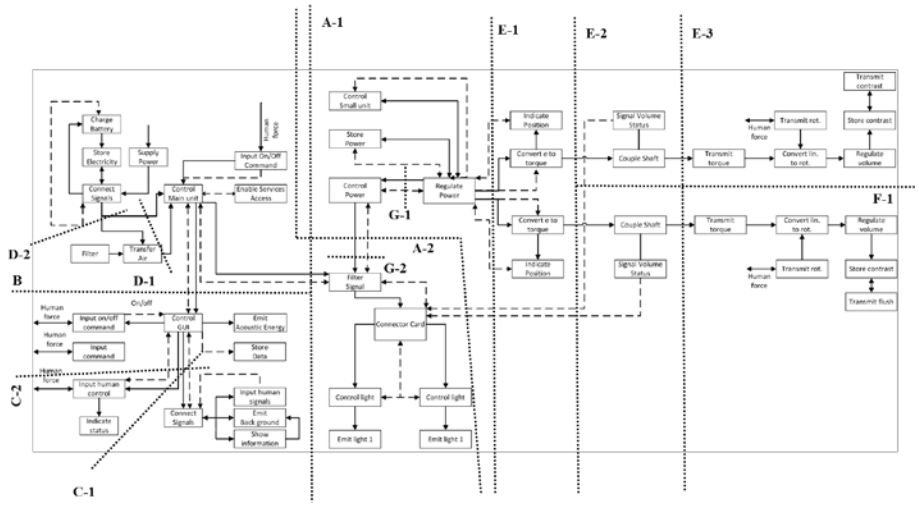

Figure 6.1. MRI Machine Injector Function Structure with different bi-sectioning cuts suggested by the algorithm in second approach

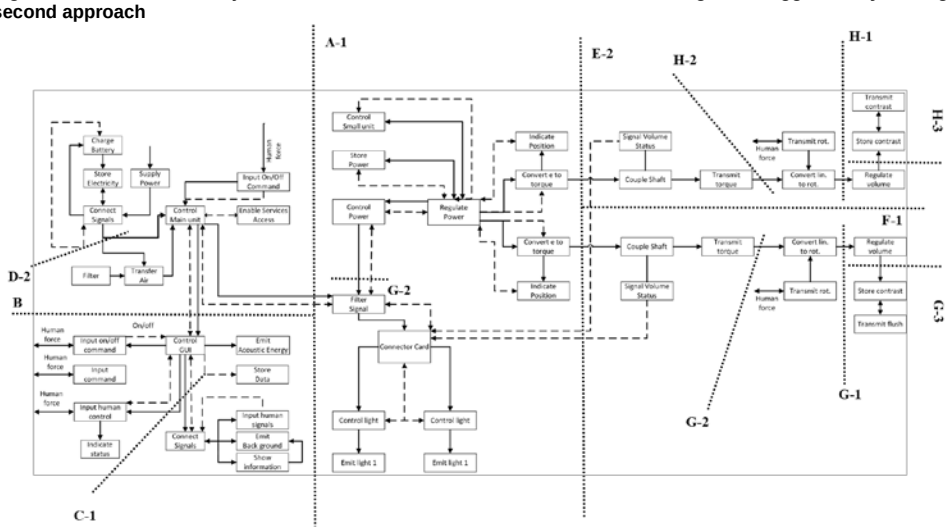

Figure 6.2. MRI Machine Injector Function Structure with different bi-sectioning cuts selected by designer in second approach to incorporate modularity constraints

In the third approach, the initial algorithm-proposed clustering scheme is illustrated in Figure 7.1. This architecture violates both the magnetic and sterility related constraints, so similar to the previous approaches the *couple shaft* and *transmit torque* ~~fucntions~~functions are constrained to be in the same module. The algorithm then suggests a new optimal architecture given the imposed constraints, as shown in Figure 7.2. One module expands to compensate the part lost by separation of the two modules and to reduce the number of module boundaries. Next, adding sterility constraints a new module with the *transmit contrast/flush* and *store contrast* functions is suggested by the algorithm. The accepted clustering is shown in Figure 7.3.

There is a notable difference between our first and third approach. In the first approach, the designer determines the modules and the algorithm merely informs the designer about outcomes. In the third approach, the algorithm clusters the elements, and after every outcome, the designer judges the result and provides added constraints determined to be violated.

The second approach fits between the two in terms of designer control over product architecture. In this particular example, all three approaches resulted into the same example. This is not the case in general, particularly for more complex examples where there are more alternatives and paths of decisions.

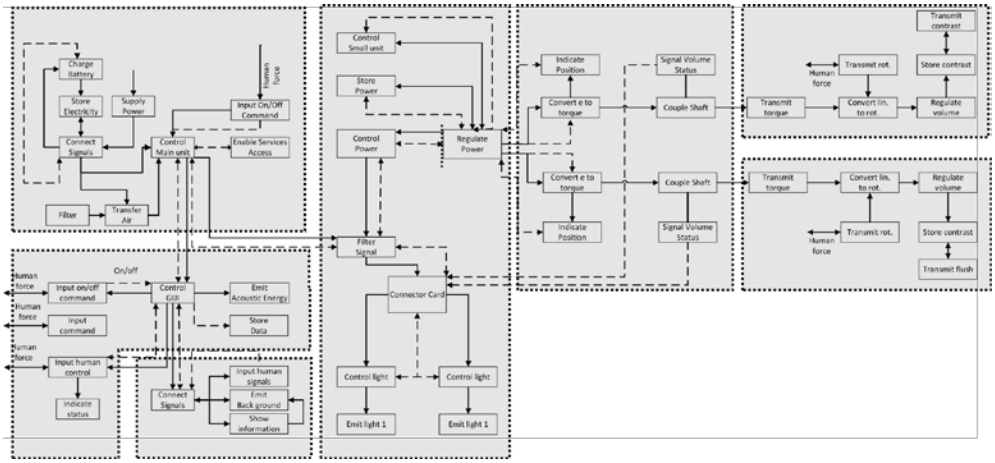

Figure 7.1. MRI Machine Injector function structure with initial optimal structure suggested by the algorithm

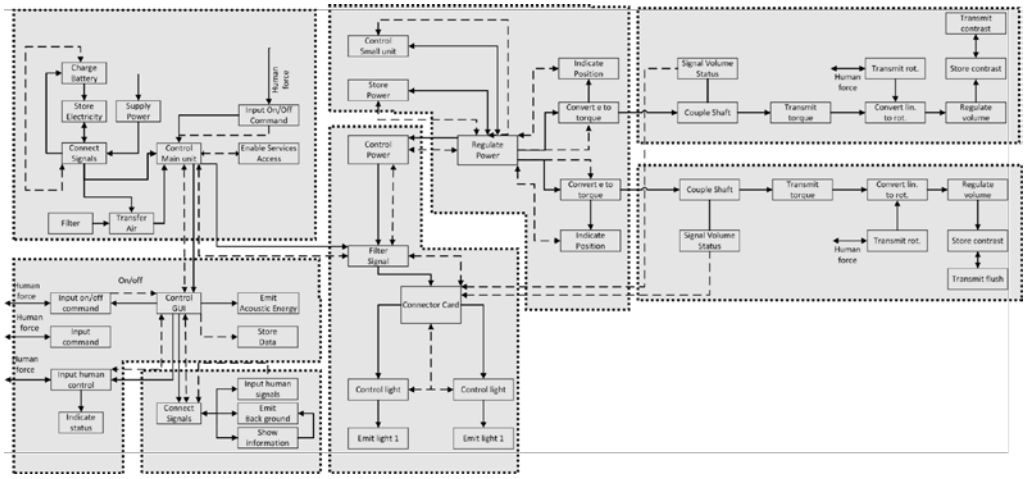

Figure 7.2. MRI Machine Injector function structure with optimal structure produced by the algorithm after imposing electromagnetic field related constraints.

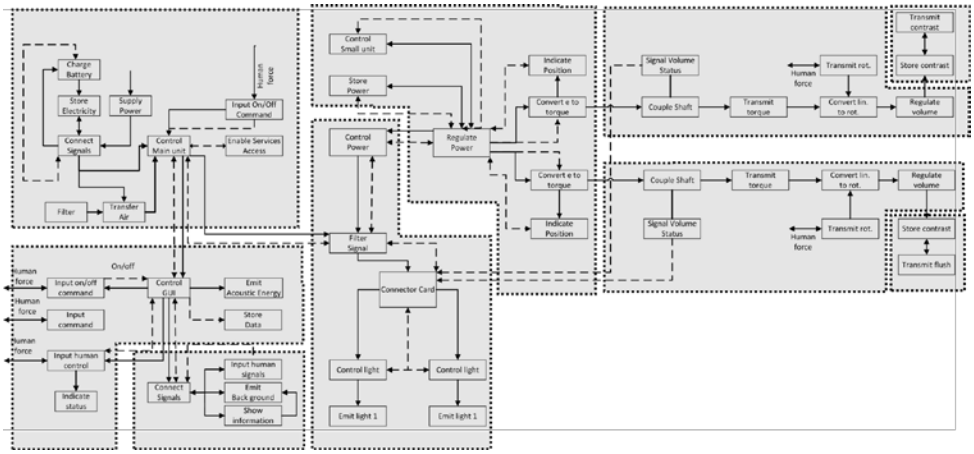

Figure 7.3. MRI Machine Injector function structure with optimal structure produced by the algorithm after imposing electromagnetic field related constraints.

LIMITATIONS

At this point we have demonstrated that the purposed algorithms can be used with several approaches to incorporate architecting constraints that otherwise would not be considered in conventional clustering algorithms. Yet, we have not yet fully examined these mechanisms in detailed user studies of designers, to see how they might really use the tools and the potentials and pitfalls in practice. We also have not studied the user interface and the form of communication between the algorithm and designer. For example in the first approach, we can display a value at the top of each flow in a schematic diagram to indicate how strongly the algorithm recommends it as a module interface, or we can alternatively use rankings, or other indicators such as the effect on different modularity metrics. This is all future research.

Beyond user studies, the first and second approach requires an initial estimate of what a "good" architecture is, based on a particular chosen metric. In practice any architecture might be better attributed a level of appropriateness rather than only "good" or not. Another avenue to improve the algorithm is to assign a utility score to each architecture according to the metrics and computed rankings to make suggestions.

Finally, as the number of elements increases, generating all feasible architectures becomes computationally intensive quickly and it is not be feasible for real-time interactive sessions. Research is needed on more efficient algorithms and pre-computing possible alternatives.

CONCLUSION

We have shown cooperative modularization using both algorithms and designer input, and by doing so we can incorporate architectural constraints which generally could not be considered previously. We ~~purposed~~ propose three different approaches to interacting with a clustering algorithm, each with slight modifications to the algorithm. In the first approach, a designer determines the connections to form the module interfaces, and the algorithm ranks a next set of connections for potential module boundaries. In the second approach the designer iteratively splits the schematic into sections, and the algorithm ranks the next set of different feasible divisions based on their effect on the metrics. In the third method, the algorithm clusters modules iteratively subject to constraints that the designer considers as violated. We detailed these three approaches with an example and demonstrated their application for taking field-effect related constraints into product architecture design of a MRI injector.

ACKNOWLEDGEMENT

The authors would like to thank the SUTD-MIT International Design Centre (IDC, idc.sutd/edu/sg) for financial and intellectual support. Any opinions, findings, or recommendations are those of the authors and do not necessarily reflect the views of the IDC.

REFERENCES

[1] Du, X., Jiao, J., and Tseng, M. M., 2001, "Architecture of product family: fundamentals and methodology," Concurr. Eng., **9**(4), pp. 309–325.

[2] Gershenson, J. K., Prasad, G. J., and Zhang, Y., 2003, "Product modularity: definitions and benefits," J. Eng. Des., **14**(3), pp. 295–313.

[3] Marshall, R., and Leaney, P. G., 1999, “A systems engineering approach to product modularity,” Proc. Inst. Mech. Eng. Part B J. Eng. Manuf., **213**(8), pp. 847–851.
[4] Hölttä, K. M., and Otto, K. N., 2005, “Incorporating design effort complexity measures in product architectural design and assessment,” Des. Stud., **26**(5), pp. 463–485.
[5] Dahmus, J. B., Gonzalez-Zugasti, J. P., and Otto, K. N., 2001, “Modular product architecture,” Des. Stud., **22**(5), pp. 409–424.
[6] Sanchez, R., and Mahoney, J. T., 1996, “Modularity, flexibility, and knowledge management in product and organization design,” Strateg. Manag. J., **17**(S2), pp. 63–76.
[7] Stone, R. B., Wood, K. L., and Crawford, R. H., 1998, “A heuristic method to identify modules from a functional description of a product,” Proceedings of DETC98, pp. 1–11.
[8] Hirtz, J., Stone, R. B., McAdams, D. A., Szykman, S., and Wood, K. L., 2002, “A functional basis for engineering design: reconciling and evolving previous efforts,” Res. Eng. Des., **13**(2), pp. 65–82.
[9] Stone, R. B., and Wood, K. L., 2000, “Development of a functional basis for design,” J. Mech. Des., **122**(4), pp. 359–370.
[10] Zamirowski, E. J., and Otto, K. N., 1999, “Identifying product family architecture modularity using function and variety heuristics,” 11th International Conference on Design Theory and Methodology, ASME, Las Vegas.
[11] Sharman, D. M., and Yassine, A. A., 2007, “Architectural valuation using the design structure matrix and real options theory,” Concurr. Eng., **15**(2), pp. 157–173.
[12] Suh, E. S., and Kott, G., 2010, “Reconfigurable parallel printing system design for field performance and service improvement,” J. Mech. Des., **132**(3), p. 034505.
[13] Chiriac, N., Hölttä-Otto, K., Lysy, D., and Suh, E. S., 2011, “Level of modularity and different levels of system granularity,” J. Mech. Des., **133**(10), p. 101007.
[14] Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., and Clarkson, P. J., 2009, “Change propagation analysis in complex technical systems,” J. Mech. Des., **131**(8), p. 081001.
[15] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2003, “Identifying modular and integrative systems and their impact on design team interactions,” J. Mech. Des., **125**(2), pp. 240–252.
[16] Sanaei, R., Otto, K., Hölttä-Otto, K., and Luo, J., 2015, “Trade-Off Analysis of System Architecture Modularity Using Design Structure Matrix,” ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. V02BT03A037–V02BT03A037.
[17] Hölttä-Otto, K., Chiriac, N. A., Lysy, D., and Suk Suh, E., 2012, “Comparative analysis of coupling modularity metrics,” J. Eng. Des., **23**(10-11), pp. 790–806.
[18] Otto, K., and Hölttä-Otto, K., 2010, “Incorporating field effects into modular architecture methods,” ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 543–551.
[19] Sudjianto, A., and Otto, K., 2001, “Modularization to support multiple brand platforms,” Proc. ASME International Design Engineering Technical Conferences, Design Theory and Methodology.
[20] Dahmus, J. B., and Otto, K. N., 2001, “Incorporating lifecycle costs into product architecture decisions,” Proc 2001 ASME Design Engineering Technical Conferences.
[21] Pahl, G., and Beitz, W., 2013, Engineering design: a systematic approach, Springer Science & Business Media.
[22] McAdams, D. A., Stone, R. B., and Wood, K. L., 1999, “Functional interdependence and product similarity based on customer needs,” Res. Eng. Des., **11**(1), pp. 1–19.
[23] Schaeffer, S. E., 2007, “Graph clustering,” Comput. Sci. Rev., **1**(1), pp. 27–64.
[24] Ding, C. H., He, X., Zha, H., Gu, M., and Simon, H. D., 2001, “A min-max cut algorithm for graph partitioning and data clustering,” Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on, IEEE, pp. 107–114.
[25] Fortunato, S., 2010, “Community detection in graphs,” Phys. Rep., **486**(3), pp. 75–174.
[26] Garey, M. R., Johnson, D. S., and Stockmeyer, L., 1976, “Some simplified NP-complete graph problems,” Theor. Comput. Sci., **1**(3), pp. 237–267.
[27] Anderberg, M. R., 2014, Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks, Academic press.
[28] Newman, M. E., and Girvan, M., 2004, “Finding and evaluating community structure in networks,” Phys. Rev. E, **69**(2), p. 026113.
[29] Yu, T.-L., Yassine, A. A., and Goldberg, D. E., 2003, “A genetic algorithm for developing modular product architectures,” ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 515–524.
[30] Wang, B., and Antonsson, E. K., 2004, “Information measure for modularity in engineering design,” ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, pp. 449–458.
[31] Duch, J., and Arenas, A., 2005, “Community detection in complex networks using extremal optimization,” Phys. Rev. E, **72**(2), p. 027104.
[32] Bui, T. N., and Moon, B. R., 1996, “Genetic algorithm and graph partitioning,” Comput. IEEE Trans. On, **45**(7), pp. 841–855.

[33] Hansen, P., and Jaumard, B., 1997, “Cluster analysis and mathematical programming,” Math. Program., **79**(1-3), pp. 191–215.
[34] Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C., 1989, “Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning,” Oper. Res., **37**(6), pp. 865–892.
[35] McSherry, F., 2001, “Spectral partitioning of random graphs,” Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on, IEEE, pp. 529–537.
[36] Al-Sultan, K. S., 1995, “A tabu search approach to the clustering problem,” Pattern Recognit., **28**(9), pp. 1443–1451.
[37] Mattson, C. A., and Magleby, S. P., 2001, “The influence of product modularity during concept selection of consumer products,” Proceedings of the ASME DETC Design Theory and Methodology Conference, Pittsburgh, PA, Sept, pp. 9–12.
[38] Aarnio, J., 2003, Modularization by Integration: Creating Modular Concepts for Mechatronic Products, Tampere University of Technology.
[39] Holtta, K. M., and Salonen, M. P., 2003, “Comparing three different modularity methods,” ASME 2003 international design engineering technical conferences and computers and information in engineering conference, American Society of Mechanical Engineers, pp. 533–541.