Proceedings of ICAD2011
The Sixth International Conference on Axiomatic Design
Daejeon – March 30-31, 2011
ICAD-2011-02

AXIOMATIC MODULAR SYSTEM DESIGN FOR SERVICE-ORIENTED PRODUCTS

Eli Stiassnie
elisti@technion.ac.il
Laboratory for CAD and LCE
Faculty of Mechanical Engineering
Technion - Israel Institute of Technology
32000 Haifa, Israel

Moshe Shpitalni
shefi@tx.technion.ac.il
Laboratory for CAD and LCE
Faculty of Mechanical Engineering
Technion - Israel Institute of Technology
32000 Haifa, Israel

ABSTRACT

The business solution to the three-sided conflict among industry, customers and the environment is based upon transitioning from the selling of products to the selling of services, while the manufacturer remains the owner of the product and is responsible for it throughout its lifecycle. A key to the success of such a transition is the concept of extended maintenance based upon modular product design. With the aid of axiomatic design, the product's concept can already be optimized at the preliminary design stage. Furthermore, axiomatic design leads to a substantial decrease in the product's concept complexity, that is, in the core of modular design. A system that automatically modularizes a product design can be realized by integrating principles of axiomatic design with two new design structure matrices (DSM). The goal of the proposed methodology is to automatically determine the number of modules and to identify product modules in a design, which are essential for the implementation of a module-based product-service plan. An example of automatic modularization for the design of a water dispenser is presented.

Keywords: Axiomatic Design, DSM, modularization

1 INTRODUCTION

In recent years, protecting the environment and its natural resources for future generations has become a major concern. Growing industrialization, depleted resources, population growth and globalization have prompted intensive legislation supporting a greener society. One of the consequences of "green" legislation is the need to develop and implement new tools, methodologies and approaches for designing, manufacturing and maintaining products. The leading trend gradually being implemented by manufacturers to meet both environmental and economic goals is the Selling-of-Services approach [Shpitalni, 2004].

By transitioning to selling services rather than products, the manufacturer remains the owner of the product throughout its lifecycle and is responsible for providing reliable services to the customer. As owner of the product, the manufacturer is allowed to augment lifespan-prolonging activities such as servicing and upgrades in order to provide reliable services. A major consequence of this approach is the change in product design towards efficient maintenance and end-of-life activities, such as product reclamation, disassembly and recycling [Shpitalni, 2004; Franke and Seliger, 2002].

The shift to a service-oriented business model has been driven by environmental requirements and legislation and can be economically justified only when environmental legislation is enforced. The exertion of careful control over lifecycle-enhancing activities obtained by this strategy offers substantial benefits by extending product life span, fostering exploitation of resources and reducing material and energy waste, consequently contributing significantly to the sustainability of both the product and the environment.

Implementing the product-service approach dictates new design considerations. In particular, product maintenance [Takata *et al.,* 2004] becomes more than merely a scheduled service. Maintenance — whether time- or condition-based — becomes a tool to enhance a product's active life span and reliability (which is now the interest of the manufacturer) by means of servicing, part replacement (possibly reuse of parts) and upgrade operations. Nevertheless, growing product complexity (mainly due to ongoing technological advancements and increases in interdisciplinary component integration) has made maintenance planning more difficult. Manufacturers' efforts to gain more control over product performance together with the need to provide reliable operation throughout the entire life span of the product are dictating a change in product design approaches.

Adopting a modular system architecture design has many advantages for service-oriented products, including quick and efficient maintenance and upgrades and a reduction in on-site service time needed to isolate a malfunctioning element. This lowers down-time by replacing entire modules. Although premature replacement of functioning inner-module elements does incur cost, it in turn reduces the costs of unscheduled maintenance activities, which in many cases are more substantial. Moreover, initiated replacement provides an opportunity to salvage parts, which are then refurbished and reused or recycled, consequently reducing environmental loads.

As products become large-scale and more complex, abstract models must be developed to deal with complex product architectures [Sharman and Yassine, 2004]. The abstraction in product architecture gained by modularization may aid designers in attaining a more manageable structure of the product-to-be.

The method introduced in this paper aims to support the selling-of-services paradigm. More specifically, we present a method and tool developed to support modular system

architecture design for service-oriented products. The proposed design methodology combines Axiomatic Design (AD) principles with Design Structure Matrix (DSM) analysis tools to facilitate desired product functionality as well as modular architecture (Figure 1).

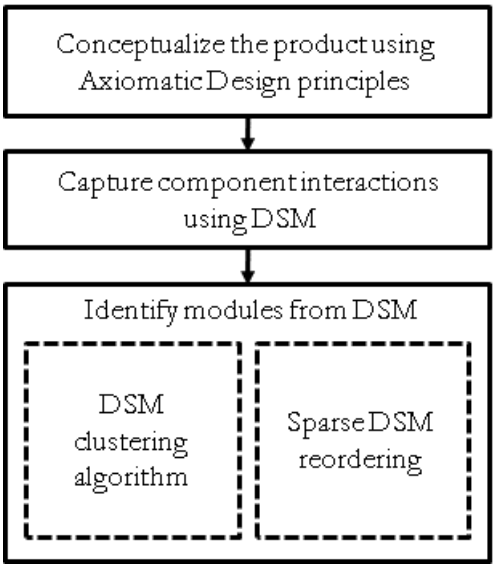


Figure 1. Proposed methodology for product modularization.

2 AIM AND METHOD

Based on the components comprising a product, the aim is to develop a system that automatically determines the optimal number of modules and integrates the components in the modules according to specific requirements related to functionality, such that the interconnections among modules are minimized. Minimizing the interactions among the different modules reduces system complexity and produces a module-oriented architecture.

The first step consists of conceptualizing the product/system using axiomatic design principles. This practice ensures maximal decoupling of design parameters and minimizes overall component interaction. Due to the nature of this process, the resulting design at this stage may already achieve some degree of component integration, laying the foundation for module identification in the following step.

The next step involves capturing structural interactions among different product components in a design structure matrix (DSM) representation based on the axiomatic design. The matrices are then used to determine the system's modulation through component clustering.

Since manual module identification by means of manual clustering is only possible for simple products, a supporting tool is needed for more intricate systems. Hence, as part of this research, two new DSM clustering tools have been developed: a new DSM clustering algorithm and a new reordering algorithm of sparse DSMs.

The resulting methodology is verified and demonstrated here on a simple case study – a water dispenser.

3 CONCEPTUALIZING THE PRODUCT THROUGH AXIOMATIC DESIGN PRINCIPLES

In engineering design, conceptual design is the stage in which the working principles and structure of the product are conceived. This process results in a primary solution, which may be considered the most suitable alternative for a given set of specifications. The process of generating the optimal concept alternative depends greatly on the methodology and the evaluation measures adopted by the product designer and can dictate product architecture and modularity.

At this stage of the proposed methodology, we consider conceptualizing the product by applying the axiomatic design methodology (ADM) [Suh, 2001]. This methodology yields a product design with the following features: (a) either a decoupled or an uncoupled design embodiment, resulting in a decrease in the number of inter-component interactions, and (b) a certain degree of component integration, which may ease the subsequent process of product modularization.

In ADM, according to the independence axiom a good design is attained only if the design is either uncoupled or decoupled [Suh, 2001], ensuring that functional requirements are independently satisfied by the corresponding design parameters.

The design matrix characterizes the product design through the relations between the FR vector (functional requirements – or goals) and the DP vector (design parameters – or solutions). The axiomatic design methodology includes three types of design matrices: coupled, uncoupled and decoupled. In the following equations, the relationships between FRs and DPs are represented either by "X" (a relationship exists) or by "0" (no relationship exists). The relationships represented by "X" are either linear (where "X" represents a constant) or nonlinear (where "X" represent functions of DPs).

Equation 1 illustrates a coupled design. This design is considered to be unacceptable, since it is hard to control a particular FR through its corresponding DP without affecting other FRs.

$$\begin{Bmatrix} FR1 \\ FR2 \\ FR3 \\ FR4 \end{Bmatrix} = \begin{bmatrix} X & X & 0 & X \\ 0 & X & 0 & X \\ X & 0 & X & 0 \\ 0 & X & X & X \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \\ DP3 \\ DP4 \end{Bmatrix} \quad (1)$$

The ideal or uncoupled design, as illustrated in equation 2, is represented by a diagonal design matrix. In this case, each of the FRs is satisfied independently by one DP.

$$\begin{Bmatrix} FR1 \\ FR2 \\ FR3 \\ FR4 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 & 0 \\ 0 & X & 0 & 0 \\ 0 & 0 & X & 0 \\ 0 & 0 & 0 & X \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \\ DP3 \\ DP4 \end{Bmatrix} \quad (2)$$

In a case where the design can be represented by a triangular design matrix (Equation 3), a decoupled (or acceptable) design is achieved. In this case, FR independence can be guaranteed by determining the proper sequence of manipulating DPs.

$$\begin{Bmatrix} FR1 \\ FR2 \\ FR3 \\ FR4 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 & 0 \\ 0 & X & 0 & 0 \\ X & 0 & X & 0 \\ X & X & X & X \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \\ DP3 \\ DP4 \end{Bmatrix} \quad (3)$$

For a system to achieve modularity, the physical units or modules must be distinguishable and separated into units (with minimum interaction and/or relations between units-interdependencies). Separation into modules is only possible if elements in the physical design are unaffiliated to some extent. Although only functional independence is practically achieved, the second axiom suggests that physical integration is desirable between elements, reducing product complexity (information content) and resulting, in many cases, in a modular-like form.

By applying axiomatic design principles at the conceptual phase, we can to some degree ensure that physical elements are integrated into a single module, thus reducing overall system complexity and maximizing potential modularity.

4 CAPTURING STRUCTURAL INTERACTIONS USING DESIGN STRUCTURE MATRICES

In this stage of the proposed method, all adjacencies, interactions and integrations among the product components are modeled based on the conceptual design attained in the previous stage. These interactions and integrations include architectural/spatial dependencies and process flows, as well as material, information and energy flows. These are identified by developing graphical representations of the interdependent flows and then by developing the consequent design structure matrix (DSM) for each type of interaction.

DSM is a system engineering tool that uses matrices to model and analyze complex projects, processes or systems [Browning, 2001] DSMs capture the structure of the interactions, interdependencies and interfaces among different hierarchical product elements (i.e., components and modules), as shown in Figure 2. Typically DSM analysis uses clustering algorithms to identify groups of close-coupled components, as shown in Figure 3. The primary objective of clustering elements into modules is to minimize interactions among different modules that reduce system complexity and lead to module-oriented architecture.

Analyzing system modularity through DSM involves the following three steps:

1. Identifying the system components based on the conceptual design.
2. Defining all adjacencies, interactions and integrations between the components.
3. Analyzing potential clustering of system components (integration analysis).

Integration analysis is a tool that offers insights into possible system modularization by clustering off-diagonal elements and reordering rows and columns of the DSM elements. The foremost objective of clustering is to minimize interactions among different clusters, sub-systems or modules.

To date, several algorithms and heuristics have been proposed for clustering [Sanchez and Mahoney, 1997; Fernandez 1998], but no single clustering approach has been identified to give an optimal solution. Therefore, as part of this research and methodology, we have tailored two independent tools for Design Structure Matrix reordering and clustering. The first is a DSM clustering algorithm and the second is an alternative new clustering approach for sparse DSMs, based on sparse reverse Cuthill-McKee ordering [George and Liu, 1981].

	1	2	3	4	5	6	7	8	9	10	11	12
1	*	0	2	0	2	0	0	2	0	0	0	0
2	0	*	0	0	1	2	0	0	0	1	1	0
3	2	0	*	0	1	0	1	1	0	0	0	0
4	0	0	0	*	0	0	0	0	1	1	0	0
5	2	1	1	0	*	0	0	1	0	0	0	0
6	0	2	0	0	0	*	0	0	0	0	0	0
7	0	0	1	0	0	0	*	0	0	0	0	2
8	2	0	1	0	1	0	0	*	0	0	0	0
9	0	0	0	1	0	0	0	0	*	2	0	0
10	0	1	0	1	0	0	0	0	2	*	0	0
11	0	1	0	0	0	0	0	0	0	0	*	0
12	0	0	0	0	0	0	2	0	0	0	0	*

Figure 2. An example of a synthetic artificial DSM.

Product elements and their interactions are recorded in a square matrix. Diagonal cells represent the elements (in this case 12 elements); off-diagonal cells represent interactions between elements and their strength (0-no dependency; 1-weak dependency; 2-strong dependency). If the direction of the interactions is meaningful, the directions are recorded in an asymmetric matrix, where the location of an interaction (either over or under the diagonal) represents the direction. If there are no directional preferences, DSM(i,j)=DSM(j.i)), and the DSM is symmetrical.

	5	8	3	1	11	6	2	12	7	4	10	9
5	*	1	1	2	0	0	1	0	0	0	0	0
8	1	*	1	2	0	0	0	0	0	0	0	0
3	1	1	*	2	0	0	0	0	1	0	0	0
1	2	2	2	*	0	0	0	0	0	0	0	0
11	0	0	0	0	*	0	1	0	0	0	0	0
6	0	0	0	0	0	*	2	0	0	0	0	0
2	1	0	0	0	1	2	*	0	0	0	1	0
12	0	0	0	0	0	0	0	*	2	0	0	0
7	0	0	1	0	0	0	0	2	*	0	0	0
4	0	0	0	0	0	0	0	0	0	*	1	1
10	0	0	0	0	0	0	1	0	0	1	*	2
9	0	0	0	0	0	0	0	0	0	1	2	*

Figure 3. The DSM from Figure 2 after integration analysis; four modules were identified for integration.

5 A DSM CLUSTERING ALGORITHM FOR MODULE IDENTIFICATION

For small and sparse DSMs, manual clustering by visual inspection is often possible. But for more complex products, computer-aided clustering is essential.

The primary objective of clustering is to minimize interactions among different clusters [Browning, 2001]. As part of this research, a new heuristic clustering algorithm was developed and implemented on the MATLAB™ platform. The algorithm is based upon integrating components into modules rather than subdividing the product into modules, as proposed by Sanchez and Mahoney [1997] and by Fernandez [1998]. The proposed clustering algorithm divides the system into several clusters based on the element interactions obtained in the previous stage. The proposed clustering algorithm is based on the following objectives:

(a) The number of outer-cluster interactions needs to be minimized to reduce system complexity.
(b) Inner-cluster connectivity should include all components in the module to assure fully integrated clusters.
(c) No overlapping clusters are allowed to attain total cluster separation corresponding to disjoint modules.

Applying these objectives results in an optimal balance between inner-cluster connectivity (all element in the cluster are related) and the reduction of outer-cluster interactions. Therefore, the algorithm avoids converging to one mega-cluster by splitting clusters that are not fully connected into independent or unrelated clusters.

The main idea behind our clustering algorithm is that modules should be independent and totally separate, contrary to other heuristics [Sanchez and Mahoney, 1997; Fernandez, 1998]. This dictates that interface locations cannot be shared between clusters and overlapping between clusters is prohibited. Since interface locations are usually the weakest links in the design, they should either be defined as interface units (links) or integrated into one of the adjacent modules, if possible. Furthermore, if the design includes a high level integrating component or bus (a module or component integrating many modules together, as in a PC motherboard), this component should be excluded from the clustering process to ensure that the modules are separated.

We define a cluster as a group of components with coupled interactions among them and minimal external interactions. Consequently, in the proposed algorithm all inner-cluster elements must be connected, be part of a continual flow or have the required spatial adjacency. Thus, the objective function of our clustering algorithm is to minimize the number of interactions outside the clusters:

$$f = \min\left[DSM\ Cost\right] \tag{4}$$

where

$$DSM\ Cost = \sum_{i=1}^{n}\sum_{j=1}^{n} DSM(i,j) \tag{5}$$

$$\forall (i,j)\ \ outside\ \ clusters$$

The clustering algorithm initializes a cluster list with n clusters, each of which comprises exactly one component (n 1x1 matrices), where n is the size of the DSM. The clustering sequence randomly selects a cluster and then for every other potential component evaluates a candidate to be integrated into the selected cluster. The algorithm then selects the element that most improves the objective function f. If f cannot be improved, the algorithm returns to the previous step and selects a new candidate cluster.

To reduce the possibility of a result converging to a local minimum, the algorithm randomly allows a non-improving element to be selected. After a change has been accepted, the cluster list is updated, and the resulting clusters are checked for inner-cluster connectivity. If inner-connectivity is not found, the decomposable clusters are then split to prevent the algorithm from converging to one mega-cluster. The algorithm continues to improve f iteratively until the maximum number of iterations allowed has been carried out. To improve user control, additional constraining parameters were added, among them maximum cluster size and/or minimum number of clusters.

Figure 4 depicts some modularization stages and the convergence of the algorithm.

6 A DSM REORDERING TOOL USING SPARSE REVERSE CUTHILL-MCKEE ORDERING

Efficient separations into modules can only be achieved if the number of interactions and their coupling nature are such that the clusters are distinguishable. The basic idea behind most clustering algorithms (e.g., the algorithm proposed in the last section) is that during the reordering of elements in the DSM and the repositioning of off-diagonal interactions closer to the diagonal, clusters can often be identified, as shown in Figure 5. Assuming a certain average number of interactions (ANI) per element, we show (Figure 6) that a DSM becomes sparser as its dimensions increase. Moreover, in large DSMs the density does not change significantly with ANI. Hence, we can assume that large systems have sparser DSMs.

Taking advantage of the sparsity property, we can implement the Sparse Reverse Cuthill-McKee Ordering1 [George and Liu, 1981] (SRCMO) algorithm to reduce the DSM's bandwidth. The reduction of the DSM's bandwidth has an effect similar to clustering, whereby the non-zero elements become closer to the diagonal. The SRCMO algorithm first finds a pseudo-peripheral vertex of the graph representing the matrix. It then generates a level structure by breadth-first searching and orders the vertices by decreasing the distance from the pseudo-peripheral vertex. This property may be used for reordering and pre-processing sparse DSMs before implementing either manual or automated clustering algorithms. Note that the ordering algorithm can be applied both to symmetric and to non-symmetric DSMs, consequently reducing clustering time. Figure 8 illustrates a permutation of a DSM using SRCMO.

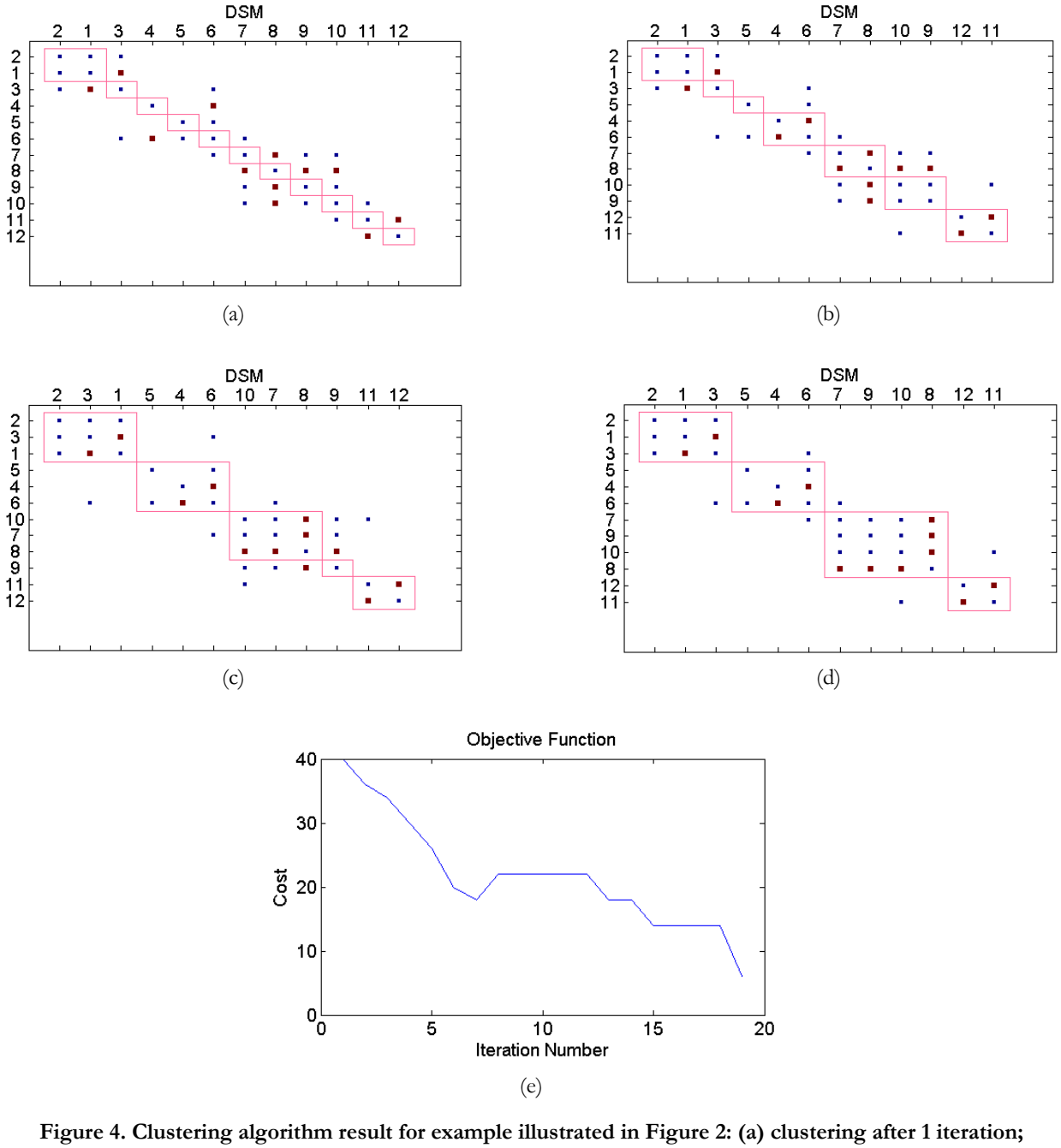


Figure 4. Clustering algorithm result for example illustrated in Figure 2: (a) clustering after 1 iteration; (b) after 5 iterations; (c) after 15 iterations; (d) after 18 iterations; (e) algorithm reached a minimum cost of 6 after 18 iterations, resulting in four clusters.

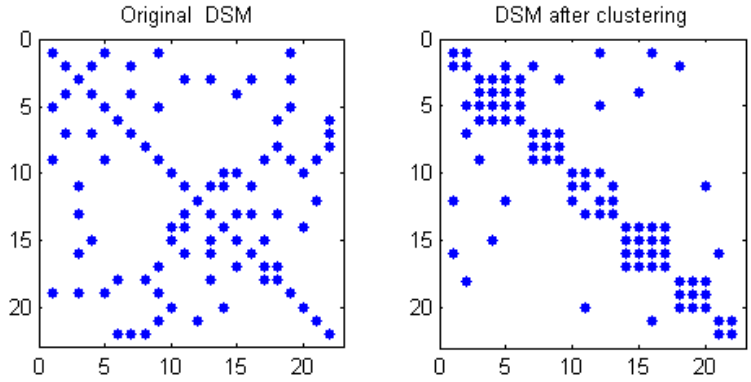


Figure 5. Clustering algorithms reposition off-diagonal interactions closer to the diagonal, creating identifiable clusters (potential modules).

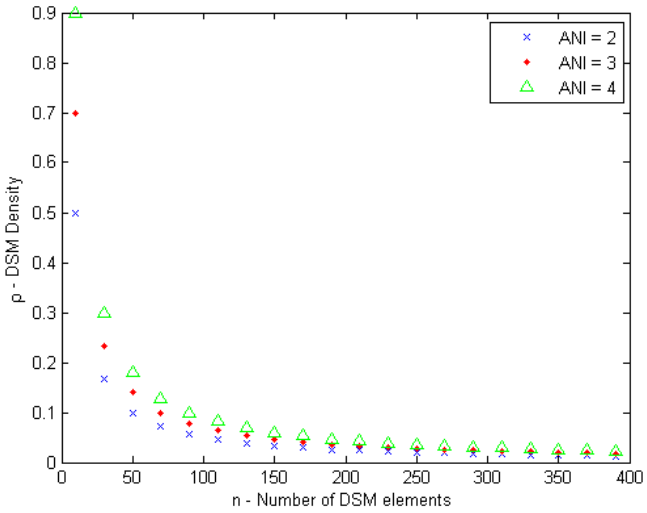


Figure 6. DSM density vs. DSM size. As the number of elements in a DSM increases, the DSM becomes sparser. For large systems, the average number of interactions per element (ANI) hardly affects DSM density.

	9	4	10	6	11	2	5	1	8	3	7	12
9	x	1	2	0	0	0	0	0	0	0	0	0
4	1	x	1	0	0	0	0	0	0	0	0	0
10	2	1	x	0	0	1	0	0	0	0	0	0
6	0	0	0	x	0	2	0	0	0	0	0	0
11	0	0	0	0	x	1	0	0	0	0	0	0
2	0	0	1	2	1	x	1	0	0	0	0	0
5	0	0	0	0	0	1	x	2	1	1	0	0
1	0	0	0	0	0	0	2	x	2	2	0	0
8	0	0	0	0	0	0	1	2	x	1	0	0
3	0	0	0	0	0	0	1	2	1	x	1	0
7	0	0	0	0	0	0	0	0	0	1	x	2
12	0	0	0	0	0	0	0	0	0	0	2	x

Figure 7. Results of applying SRCMO to the artificial example in Figure 2. In this case, SRCMO reorders the DSM so clusters are immediately recognized and no further clustering is needed.

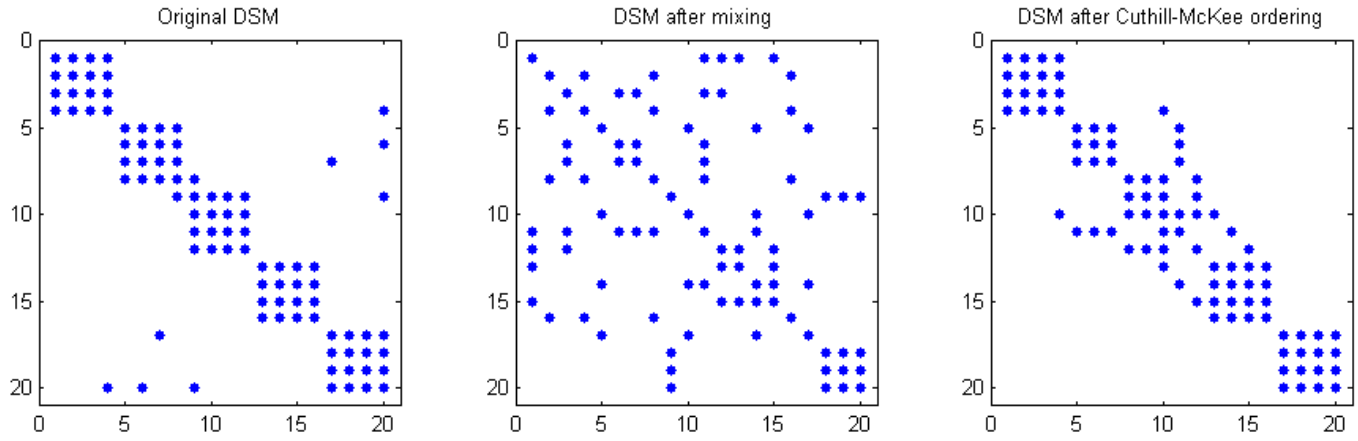


Figure 8. Permutation of a DSM using Sparse Reverse Cuthill-McKee Ordering. The clustered DSM is mixed and then processed through SRCMO. Some of the resulting clusters in the DSM can be identified.

7 CASE STUDY EXAMPLE

To demonstrate the modular design process presented in this paper, we introduce a case study: a modular design of a water dispenser. The water dispenser is designed to serve various functions, primarily to supply hot or cold filtered water in a way that is safe and sustainable.

First step: Design the product based on axiomatic design principles. After the functional requirements for the product were determined, the corresponding design parameters were conceived (Table 1). These corresponding design parameters were selected in accordance with the independence axiom to ensure at least a decoupled design. Figure 9 illustrates the results of this design process, where a decoupled design was obtained as identified by the triangular design matrix. The decoupled design ensures that the design is acceptable and minimally coupled. Furthermore, the proper sequence was determined for manipulating design parameters to gain better control of each FR separately, thus eliminating imaginary complexity.

Second step: Identify and model the product's elements and interactions based on the design. For this particular design, several interactions were considered and modeled (Figure 10), including material, heat, information, electricity and spatial (architectural) interactions. From the resulting model, the interactions were captured into DSMs. Figure 11 shows an example of a DSM map for spatial or architectural interactions; in this case a four-degree scale was adopted, including negative relations where applicable.

Third step: Identify modules using DSM clustering tools. The spatial DSM was selected for clustering, and our module-identifying clustering algorithm was applied. The results, as illustrated in Figure 12, show that the algorithm converged to a minimum after 180 iterations.

Fourth step: Applying the clustering algorithm for the water dispenser yielded five clusters representing the product modules (Figure 13). Each module was analyzed to identify its functional and architectural significance. For example, one of the clusters, "*the inlet block*", is responsible for water treatment and flow. It includes the booster pump, charcoal filter, UV filter and water valve. This case study illustrates how our methodology can be practically implemented to design a modular product. The systematic approach has successfully achieved a recognizable modular form that is identifiable and intuitive (each module has a recognizable architectural and/or functional purpose).

Table 1. Generic water dispenser axiomatic design.

Functional Requirements	Design Parameters
FR1=Supply hot boiling water safely FR11=Prevent misuse by children FR12=Heat water FR13=Ensure water boil FR14=Control hot water flow	DP1= Water Heating sub-System DP11= Safety Mechanism DP12=Boiler DP13=Extra hot button DP14=Water Tap
FR2=Supply cold water FR21= Cool water FR22=Control cold water flow	DP2= Water Cooling sub-System DP21=Heat pump DP22=Water Tap
FR3=Conserve energy FR31=Control hot water temperature FR32=Conserve heat FR33=Prevent mix of hot and cold water	DP3= Energy Conservation sub-System DP31=Thermostat DP32=Heat cycle DP33=Water Valve
FR4=Conserve water resource	DP4= Water Collection Sink
FR5=Maintain water quality FR51=Clean water FR52=Destroy bacteria	DP5= Filter system DP51=Charcoal filter DP52=ultraviolet lamp filter
FR6 = Maintain water pressure	DP6 = Booster Pump

		DP1				DP2		DP3			DP4	DP5		DP6
		DP11	DP12	DP13	DP14	DP21	DP22	DP31	DP32	DP33		DP51	DP52	
FR1	FR11	X												
	FR12		X						X					
	FR13			X										
	FR14				X					X		X	X	X
FR2	FR21					X			X					
	FR22						X					X	X	X
FR3	FR31							X	X					X
	FR32								X					
	FR33									X				
FR4											X			
FR5	FR51											X		
	FR52												X	
FR6														X

Figure 9. Axiomatic design matrix for generic water dispenser.
The result of the design process is a decoupled design.

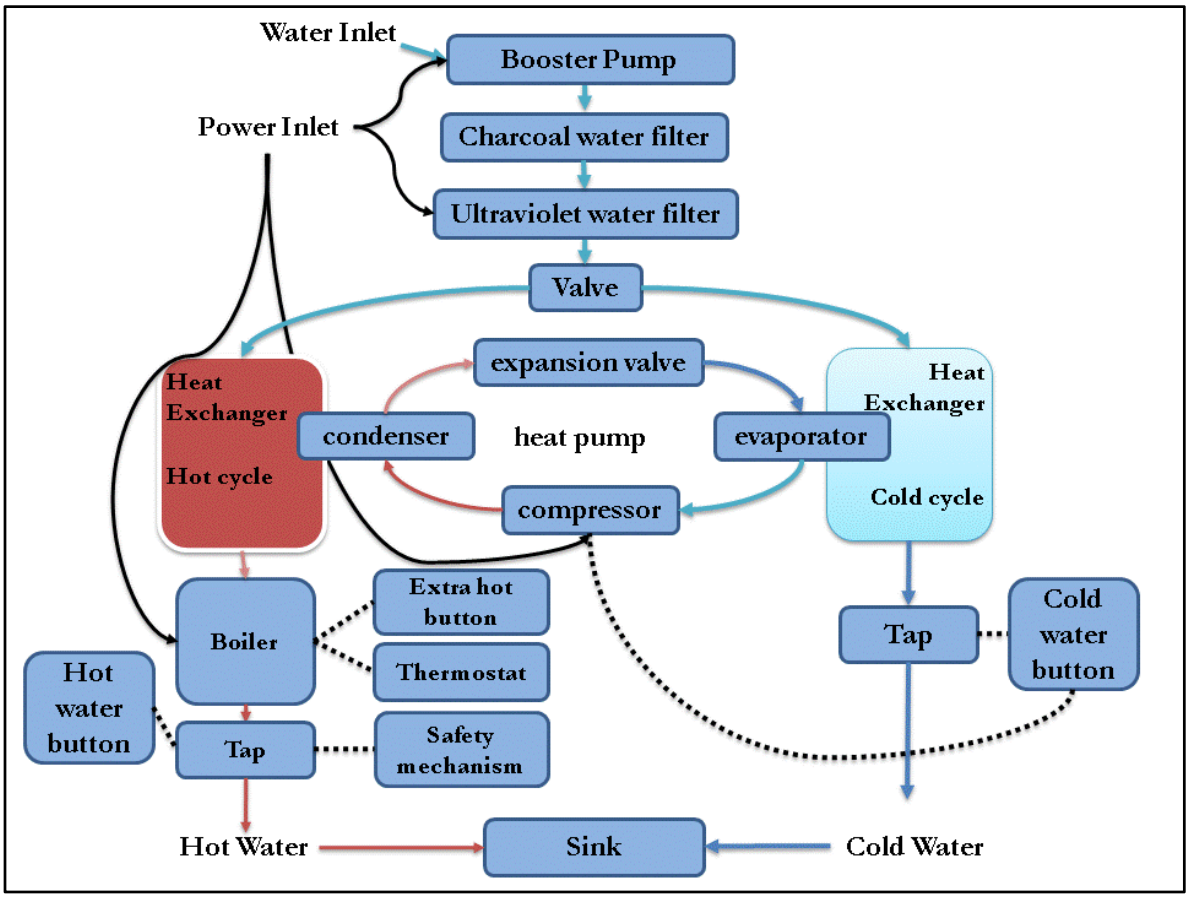
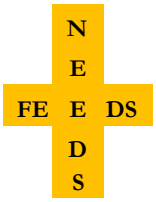


Figure 10. Inter-element interactions model of the water dispenser case study (water, heat, information, electric and spatial).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	x	0	-2	0	0	0	0	0	0	0	0.5	0	0	0	0	0	2	0	0
2	0	x	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
3	-2	0	x	0	0.5	0	-2	2	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	x	1	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0
5	0	0	0.5	1	x	0	0	0	0	0	0	0	0.5	0	0	1	0	0	0
6	0	0	0	0	0	x	0.5	0.5	0	0	0	0	0	0	0	0	0	0	0
7	0	0	-2	0	0	0.5	x	0	0.5	0	2	0	0	0	0	0	0	0	0
8	0	0	2	0	0	0.5	0	x	0.5	0	-2	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0.5	0.5	x	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	x	0	0.5	1	0	0	0	0	0	0
11	0.5	0	0	0	0	0	2	-2	0	0	x	0	0	0	0	0	0	0	0
12	0	0	0	0.5	0	0	0	0	0	0.5	0	x	1	0	2	0	0	0	0
13	0	0	0	0	0.5	0	0	0	0	1	0	1	x	0	0.5	1	0	0	0
14	0	1	0	0	0	0	0	0	0	0	0	0	0	x	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	2	0.5	0	x	0	0	0	0
16	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	x	0	0	0
17	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0	0
18	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	0.5
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	x

In Contact	2
Adjoining	1
In Proximity	0.5
Far	-2



1 Boiler	11 Hot Cycle Heat Exchanger
2 Charcoal Water Filter	12 Hot Water Button
3 Cold Cycle Heat Exchanger	13 Hot Water Tap
4 Cold Water Button	14 Booster Pump
5 Cold Water Tap	15 Safety Mechanism
6 Compressor	16 Sink
7 Condenser	17 Thermostat
8 Evaporator	18 Ultraviolet Water Filter
9 Expansion Valve	19 Valve
10 Extra Hot Button	

Figure 11. DSM map of inter-element interactions of water dispenser case study. A four-degree scale was adopted, including negative relations where applicable.

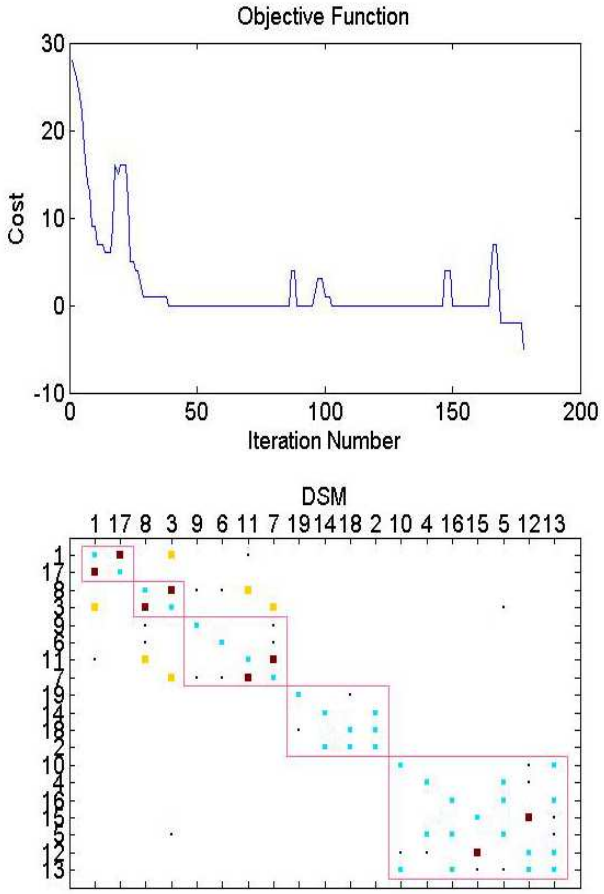


Figure 12. Clustering results for the spatial DSM of the water dispenser. Top: objective function vs. no. of iterations; bottom: the resulting clustered DSM (blue, cyan and red are positive dependency, yellow – negative dependency).

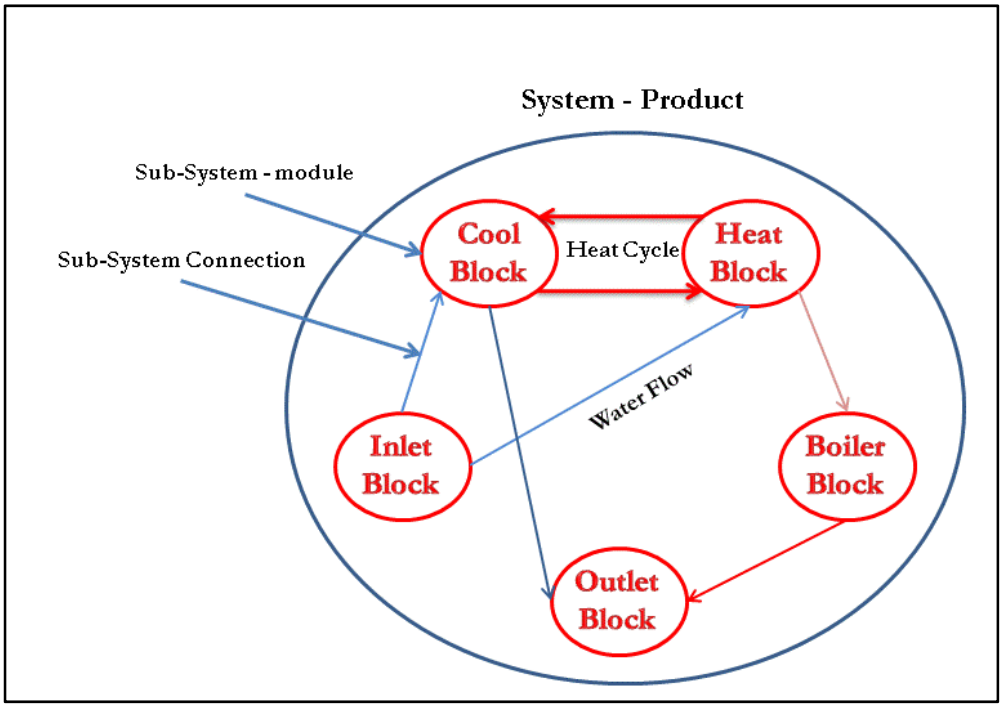


Figure 13. The five modules identified by the clustering algorithm.
The remaining outer-cluster connections represent water flow and heat exchange between modules.

8 DISCUSSION AND SUMMARY

The methodology and the complementary tools introduced in this paper support modular product design for service-oriented products. The methodology combines axiomatic design principles with design structure matrix analysis tools, thus enhancing both product functionality and modularity and yielding a structured systematic approach for identifying product modules.

The methodology comprises the following stages:

- Stage 1: Design the product based on Axiomatic Design principles to ensure (a) minimal coupling between functional embodiments, (b) improved integration between physical elements, (c) reduced system complexity and (d) maximized potential modularity.
- Stage 2: Model interdependencies and interactions among system components based on Axiomatic Design.
- Stage 3: Capture structural interactions using Design Structure Matrices based on interdependency model.
- Stage 4: Apply the clustering and/or reordering tools developed as part of this research to identify component clusters (modules/subsystems), consequently minimizing interactions among clusters and achieving a modular architecture.

A prime advantage of this approach is that both the number and the size of the modules are determined automatically and need not be decided prior to clustering. Modularization is clearly achieved by minimizing outer-cluster interactions and at the same time maintaining inter-cluster connectivity. To validate the method's ability to facilitate product modularization, we have introduced a case study. The resulting design shows that module integration is achievable and that the resulting division and distribution of the components into modules are intuitive.

9 ACKNOWLEDGMENTS

This research was funded by the Bernard M. Gordon Center for Systems Engineering at the Technion (Research Number 2009543) and supported in part by the Schlesinger Minerva Laboratory for Lifecycle Engineering and the VRL-KCiP Network of Excellence as part of the EU Sixth Framework Programme.

10 REFERENCES

[1] Browning T. R., "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions." *IEEE Transactions on Engineering Management*, 48(3), 2001. pp 292-306.

[2] Fernandez C.I.G., "Integration analysis of product architecture to support effective team co-location", Master Thesis, Massachusetts Institute of Technology, 1998.

[3] Franke C., Seliger G., "A new paradigm of manufacturing: Selling services instead of products", *Proceedings of 2002 Japan Symposium on Flexible Automation,* Hiroshima, Japan.

[4] George A., Liu J., *Computer Solution of Large Sparse Positive Definite Systems,* Prentice-Hall, 1981.

[5] Sanchez R., Mahoney J.T., "Modularity, flexibility, and knowledge management in product and organization design," *IEEE Eng. Manage. Rev.*, pp. 50–61, 1997.

[6] Sharman D.M., Yassine A.A., "Characterizing complex product architectures", *Journal of Systems Engineering,* vol. 7, pp. 35-60, 2004.

[7] Shpitalni M., "Impact of Life Cycle Approach and Selling of Services on Product Design", *Proceedings of the Manufacturing '04 Conference,* Poznan, Poland, 2004, pp. 4-5.

[8] Suh N.P., *Axiomatic design: Advances and applications*, Oxford University Press, NY, 2001.

[9] Takata S., Kimura F., van Houten F.J.A.M., Westkämper E., Shpitalni M., Ceglarek D., Lee J., "Maintenance: Changing Role in Life Cycle Management", *Annals of the CIRP,* Vol. 53, No. 2, p. 643, 2004.

[10] Umeda Y., Fukushige S., Tonoike K., "Evaluation of scenario-based modularization for lifecycle design", *Annals of the CIRP,* Vol. 58/1 p. 1-4, 2009.