KEYWORDS Clustering · Design Structure Matrix · Module Definition · Modularity · Genetic Algorithm

A Clustering Method Using New MODULARITY INDICES AND A GENETIC ALGORITHM WITH EXTENDED CHROMOSOMES

ABSTRACT

Module definition entails clustering a product architecture into independent or coordinated modules. Clustering algorithms based on Design Structure Matrices (DSMs) for defining modules have been widely studied. After reviewing existing clustering algorithms, we introduce simple new metrics that can be used as modularity indices bounded between 0 and 1 and also utilized as the objective functions to obtain optimal DSMs by maximizing interactions within modules and interactions between modules. As a search strategy for clustering modules, a combinatorial genetic algorithm using a new extended chromosome approach and modified operators for the chromosome is suggested. The module definition results indicated that the proposed clustering method using new modularity indices and genetic algorithm helps obtain optimal modular product architectures more logically.

Sangjin Jung
The Pennsylvania State University, University Park, PA, USA
sxj26@psu.edu

Timothy W. Simpson
The Pennsylvania State University, University Park, PA, USA
tws8@psu.edu

1. Introduction

Many companies are trying to not only provide customers with differentiated products on time but also reduce costs. Product planning and development strategies focused on single products can be ineffective from a product family point of view. Product planning, development, and production based on modular product platforms are known to provide a competitive strategy for reducing total cost as well as shortening time to market.

Defining modules and the modular product architecture is a key step in a modular platform definition process *(Otto et al., 2013)*. After the 1990s, various methods to define modules for product families have been widely studied *(Gershenson et al., 2004; Otto et al., 2013; Simpson, 2004)*. Particularly, many module definition methods were focused on matrix-based approaches because using a matrix such as the Design Structure Matrix *(DSM) (Browning, 2001; Eppinger and Browning, 2012; Steward, 1981)* is an effective way to represent and visualize the relationships between elements within a product architecture. In recent years, clustering algorithms using DSMs and numerical heuristic methods have been actively studied to cluster the elements within a product architecture into modules *(Borjesson and Hölttä-Otto, 2012; Helmer et al., 2010; Thebeau, 2001; Yu et al., 2003; Yu et al., 2007)*.

Most clustering algorithms utilize a measure to assess the degree of modularity, and the modularity measure is usually used as the objective function of the clustering algorithm. Guo and Gershenson *(2003)* and Gershenson et al. *(2004)* reviewed various modularity measures. Guo and Gershenson *(2004)* also introduced a modularity metric that maximize connections within a module and minimize the connectivity between modules simultaneously, but it is not suitable to use this metric directly as an objective function for clustering DSMs.

Fernandez *(1998)* and Thebeau *(2001)* developed a clustering method using the modularity metric called total coordination cost to minimize the interactions between clusters and to maximize the interactions within clusters. Borjesson and Hölttä-Otto *(2012)* improved Thebeau's clustering algorithm to obtain useful clustering results more quickly. However, the clustering algorithm of Thebeau is known to sometimes produce too many small clusters, which cannot constitute a module *(van Beek et al., 2010)*.

Hölttä-Otto and de Weck *(2007)* introduced two types of modularity indices: *(1)* the singular value modularity index *(SMI)* to capture the degree of the closeness to the diagonal of DSM and *(2)* the non-zero fraction *(NZF)* to capture the sparsity of the interrelationships between components. These modularity indices are theoretically bounded between 0 and 1 regardless of the size of DSM, but they cannot be directly used as the objective functions for clustering DSM. This is because the SMI value of a DSM is constant even though the sequence of components within the DSM is changed.

Yu et al. *(2003, 2007)* suggest a modularity measure using the minimum description length *(MDL)* principle *(Rissanen, 1978, 1999)* to minimize the information needed to describe the connectivity between modules. Helmer et al. *(2010)* improved Yu et al.'s clustering algorithm in order to overcome the deficiencies such as excessively overlapped components between clusters. The modularity measure based on the MDL principle is known to be a useful objective function for clustering DSMs to drive module definition, but the value of this measure increases with the DSM size, making it difficult to compare modularity between modular architectures of different sizes.

Meanwhile, in recent years, genetic algorithms *(GAs)* have been used for the purpose of clustering DSMs *(Helmer et al., 2010; Kamrani and Gonzalez, 2003; Whitfield et al., 2002; Yu et al., 2007)*. This is mainly because GAs can handle discrete design variables and are far more likely to find a global optimum because they widely explore the entire design space. However, existing GAs for clustering are known to reach to the optimum solution slowly because the GAs are based on binary coding and the chromosome lengths of the GAs might be extremely extended according to the number of components or the number of modules *(Borjesson and Hölttä-Otto, 2012; Yu et al., 2007)*.

After reviewing the existing modularity measures and clustering algorithms in the literature, we propose new modularity indices that overcome the deficiencies of the previous modularity measures, and a new search strategy for clustering is suggested. In this paper, Section 2 describes the modularity indices and the proposed GA using a new extended chromosome approach and modified operators for the chromosome. Section 3 presents the module definition result of a real-world product to demonstrate the effectiveness of the proposed method. Finally, Section 4 presents some conclusions.

2. Clustering Algorithm

2.1 Modularity Indices

We introduce simple new metrics that can be used as objective functions for optimizing DSMs and also utilized as modularity indices obtained between 0 and 1. The modularity index MI_1 measures the proximity of component interactions to the diagonal of a DSM:

$$MI_1 = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left[\left(-\frac{j-i}{N-1}+1\right)\frac{DSM_{ij}+DSM_{ji}}{DSM_{\max}}\right]}{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\frac{DSM_{ij}+DSM_{ji}}{DSM_{\max}}} \qquad (1)$$

where DSM_{ij} is the value of the ith row and jth column element in *DSM*; DSM_{max} is the maximum interaction value within *DSM*; and *N* represents the total number of components. The value of MI_1 is always between 0 and 1 regardless of the size of the *DSM*. As shown in **Figure 1***(a)*, a MI_1 value close to 1 denotes that the interactions are near the diagonal of the *DSM*. Thus, MI_1 should be maximized to be close to 1 when MI_1 is utilized as an objective function for clustering *DSMs*. By doing so, the *DSM* clustered maximizing MI_1 shows which components should be arranged around a component within a module. Also, one can know which components are placed near the module boundary. For example, if there exists the interaction between two components contained within two different modules respectively, then the two components are arranged near the boundaries of each module so that they can be placed near each other. The sequence information of components within each module can also be utilized to architect product families that contain shared components and modules based on this clustering.

MI_2 represents a modularity index for the independency of each module. MI_2 is formulated as follows:

$$MI_2 = \frac{\sum_{i=1}^{m-1}\sum_{j=i+1}^{m}\left(\frac{CS_{ij}^{out}}{CS_i^{in}}+\frac{CS_{ij}^{out}}{CS_j^{in}}\right)}{m(m-1)}+\varepsilon(N-m) \qquad (2)$$

where CS_i^{in} is the sum of interaction values within ith modules; CS_{ij}^{out} is the sum of interactions between ith and jth modules; *m* is the number of modules; and ε is a very small positive number. As shown in **Figure 1***(b)*, MI_2 is usually bounded between 0 and 1. MI_2 closer to 0.0 indicates a higher degree of the module independency. In **Figure 1***(b)*, a large dot within a *DSM* represents a greater interaction value than a small dot. In some cases, MI_2 might be greater than 1 if the sum of the interaction values between modules is greater than the sum of the interactions between components within a module. In Equation *(2)*, $\varepsilon(N-m)$ is utilized to obtain the clustering result including more divided modules among the clustering results with the same MI_2 values when the proposed modularity indices are used as the objective functions for optimizing *DSMs*. ε is usually set as a sufficiently small positive number such as 10^{-8}.

The proposed modularity indices can be utilized as the objective functions when optimizing DSMs. Since we should maximize MI_1 and minimize MI_2 simultaneously to reach an optimized clustering result, a scalarization function was employed to formulate the optimization problem as follows:

$$f = \frac{MI_2}{MI_1} \qquad (3)$$

By using Equation *(3)* as the fitness function of the proposed GA, we can obtain optimal DSMs including the maximized interactions within modules and the minimized interactions between modules.

2.2 Genetic Algorithm

This section introduces a search strategy using a genetic algorithm *(GA)* for clustering DSMs.

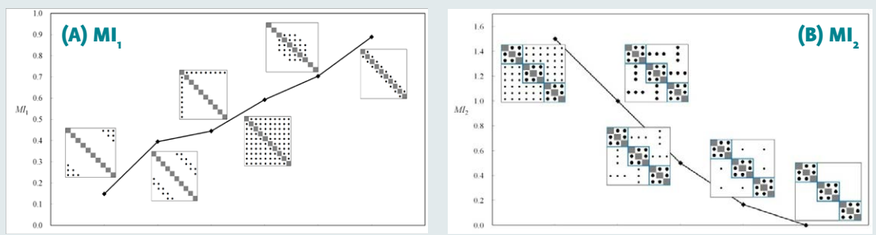


FIGURE 1. *MI_1 and MI_2 values for interaction patterns of DSMs*

To reduce the computation time of the GA, we developed a GA using an extended chromosome approach and modified operators for the chromosome.

In this study, the extended direct encoding chromosome, which can hold two kinds of information in a single chromosome, is defined based on a method suggested by Jung et al. *(2013)*. We modified their chromosome definition method in order for each chromosome to have the same string size. The extended chromosome consists of two parts. The first part of the chromosome includes the information on the sequence of each component, and the second part holds the information on the dividing points between modules. The string size of the first part is *N*, and the size of the second part is *N-1* because the number of modules can be changed from 1 to *N* during the clustering process. For example, as shown in **Figure 2**, A–H represent each component, while 3 and 6 represent the dividing points between the modules. Since the first and second dividing points are 3 and 6, respectively, the first, second, and third modules include elements A to C, elements D to F, and elements G and H, respectively.

In the proposed GA, the three operation processes of selection, cross-over, and mutation are conducted repeatedly. First, a deterministic selection *(Bäck, 1996)*, which is one of the selection methods used in existing GAs, was employed.

As for the cross-over, two kinds of cross-over methods are applied simultaneously to the two parts of the extended chromosome. The subtour-chunking cross-over *(Gen and Cheng, 1997; Grefenstette, 1987)* operates in the first part of the chromosome, and a modification of the one-cut-point cross-over *(Gen and Cheng, 1997)* is used in the second part of the chromosome. Specifically, the subtour-chunking cross-over can generate more mixed offspring as well as maintain similarly the original positions of each chunk selected from parents. The subtour-chunking cross-over method is known to give better results compared to other kinds of direct encoding cross-overs *(Gen and Cheng, 1997)*.

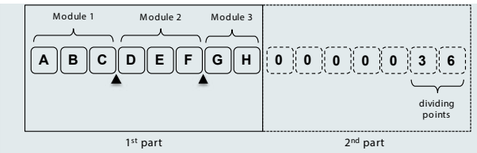


FIGURE 2. *Extended Chromosome*

An example of the proposed cross-over using the extended chromosome follows based on **Figure 3**.

1. Chunks (B C D) and (G A C B) are chosen from each first part of two parents. In the second part, the genes are jumbled.
2. The chunk (B C D) is placed into the offspring so as to maintain the original position in the Parent 1, and the chunk (G A) except the overlapped components B and C is placed to the right of the chunk (B C D).
3. The chunk (A) is chosen from the first part of Parent 1. However, the chunk is not used because (A) already exists in the offspring. An arbitrary point as the cut point of the second part is selected. The number of genes after the cut point in Parent 1 and before the cut point in Parent 2 should be the same.
4. The chunk (D F) is chosen from the first part of Parent 2, but only the chunk (F) except the overlapped component D is placed into the offspring. The cut point of the second part is selected as well.
5. The chunk (G H) is chosen from the first part of Parent 1, but only the chunk (H) is placed in the seventh position. Also, the first chunk in the second part of Parent 1 is placed into the offspring in the same position as in Parent 1.
6. The chunk (E H) is chosen from the first part of Parent 2, but only the chunk (E) is placed in the last position. Also, the first chunk in the second part of Parent 2 is spliced to the offspring.
7. The genes in the second part of the offspring are sorted in ascending order.

In the case of mutation, the swapping and altering method from *(Jung et al., 2013)* is modified and applied to the clustering problem here.

The computational procedure of the GA is as follows. In Step 1, an initial DSM is defined. In Step 2, the parameters such as mutation probability and population size are initialized. In Step 3, an initial population is generated in accordance with the given population size. In Step 4, Equations *(1)-(2)* are used to evaluate MI_1 and MI_2 for the DSM of each individual generated according to the population size. In Step 5, Equation *(3)* is employed to evaluate the fitness function *f* for the *DSM* of each individual. In Step 6, the individual with the smallest fitness function is selected, and convergence to the optimum is checked. In Step 7, the operations *(selection, cross-over, and mutation)* of the GA are performed to generate a new population, and the GA returns to Step 4.

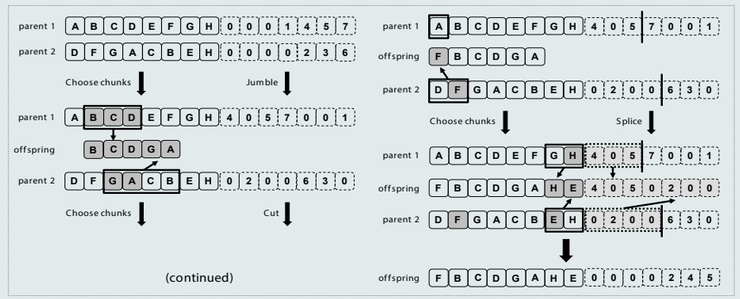


FIGURE 3. *Crossover for the extended chromosome*

3. Case Study

In order to validate the performance of the proposed method, an existing test product to define modular architecture was chosen as the case study for this paper. The product is a Black & Decker model CHV 1210 Dustbuster® *(Borjesson and Hölttä-Otto, 2012)*, which is seen in **Figure 4**. Borjesson and Hölttä-Otto *(2014)* introduced the disassembled structure of this product in detail. The Dustbuster CHV 1210 consists of 57 components *(***Table 1***)* and use 0-1 interface representation where 1 indicates a connection *(i.e., an interface exists)* and 0 indicates no connection *(Browning, 2001)*.

In this paper, the clustered DSM and the resulting modules obtained using the proposed method are compared to the clustered DSM result in reference. **Figure 5***(a)* shows the clustered DSM for the Dustbuster CHV 1210 from *(Borjesson and Hölttä-Otto, 2012)*, and **Figure 5***(b)* shows the new clustering result. As shown in **Figure 5***(a)* from the previous clustering result, the system was decomposed into too many small clusters which cannot constitute a module, even though the Dustbuster is a small electro-mechanical product. The module planning, development, manufacturing, etc. based on the defined modules containing only two or three components do not make sense. On the other hand, in the DSM clustered by the proposed method, the total number of modules is six and each module contains more components than the previous result. This new clustering result is reasonable in view of structure as shown in **Figure 4**, and each module consists of the components with the strong interactions each other.

We next evaluated the types of components contained within each module and the sequence of the components as well. For example, the newly defined module NM2 contains the components related to nozzle and filter. In the NM2, the components related to nozzle are closer together than other components and the filter-related components are closely placed each other. A similar phenomena in the sequence of components is also observed in other modules. As stated previously, the Dustbuster example just used 0-1 representation for interfaces between components in the DSM. If we utilize an advanced design dependency measure to represent interfaces, then we expect to obtain more accurate modular architecture results using the proposed method.

	(Borjesson and Hölttä-Otto, 2012)	Proposed method
Rate of active cells within modules (# of active cells within modules / # of total active cells)	69.2% (119/172)	87.8% (151/172)
MI_1	0.877	0.913
MI_2	0.099	0.066

TABLE 2. *Comparison of the rate of active cells within modules and modularity indices*

Table 2 shows a comparison of modularity index MI_1 and MI2 values and the number of active interface cells within modules. Similar to MI_1 and MI_2, the rates of active interaction cells within modules were also compared in order to check the modularity of the DSMs. Consequently, we observed that MI_1 and MI_2 values and the rate of active cells within modules of the new DSM are better than those of the previously clustered DSM as shown in **Table 2**. Thus, more components within each module were closer to the diagonal of DSM and the number of interactions between modules was also reduced as shown in **Figure 5***(b)*.

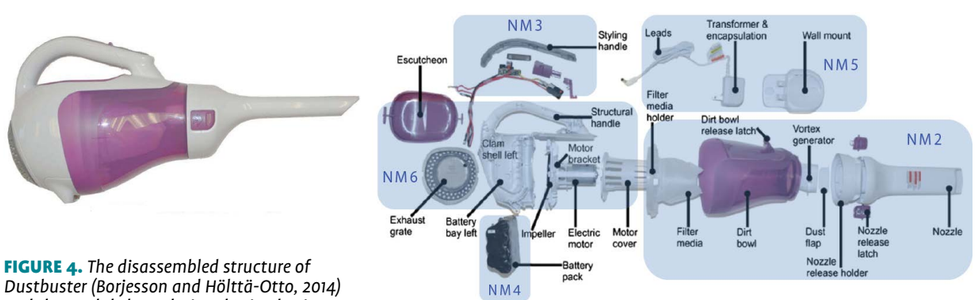


FIGURE 4. *The disassembled structure of Dustbuster (Borjesson and Hölttä-Otto, 2014) and the module boundaries obtained using the proposed method*

4. Concluding Remarks

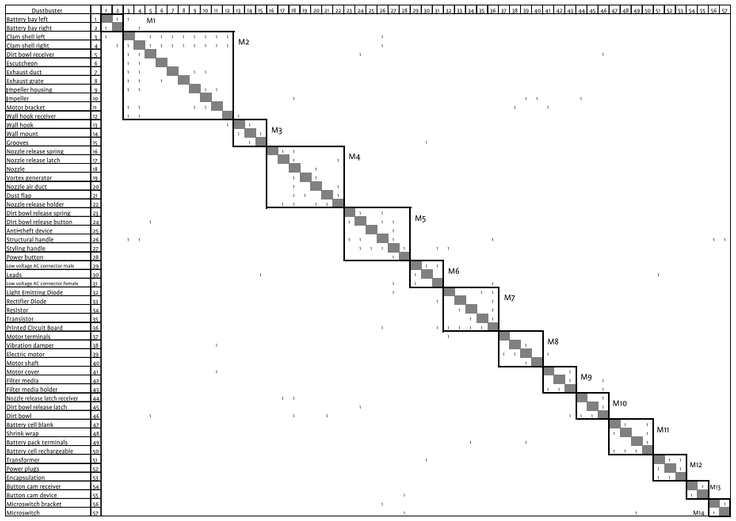
In this research, we proposed new modularity indices and investigated their impact using a new clustering method to drive module definition. The modularity index MI_1 measures the closeness to the diagonal of a DSM, and the value of MI_1 is always determined between 0 and 1 regardless of the size of the DSM. MI_2 captures the independency of each module in a clustered DSM. Thus, MI_2 closer to 0.0 indicates a higher degree of module independency. The two modularity indices were utilized as the objective function for optimizing a DSM. As a search strategy, we developed a GA using an extended chromosome, which contains two kinds of information: *(1)* the sequence of each component and *(2)* the dividing points between modules. The method for operating cross-over between the extended chromosomes was also described.

An existing test product in the literature was chosen to validate the performance of the proposed method. Comparing the new clustering result to the previous result in the literature, we observed that the MI_1 and MI_2 values of the DSMs clustered by using the proposed method are better than those of the previous DSMs. Thus, the new clustering results indicate a higher degree of the modularity on the strong interactions between components within each module *(i.e., closeness to the diagonal of DSM)* and module independency. The results also showed that module definition with the proposed method structurally yields more reasonable modular architectures. The suggested clustering method is expected to be effective for defining preliminary modules in component-based system architectures.

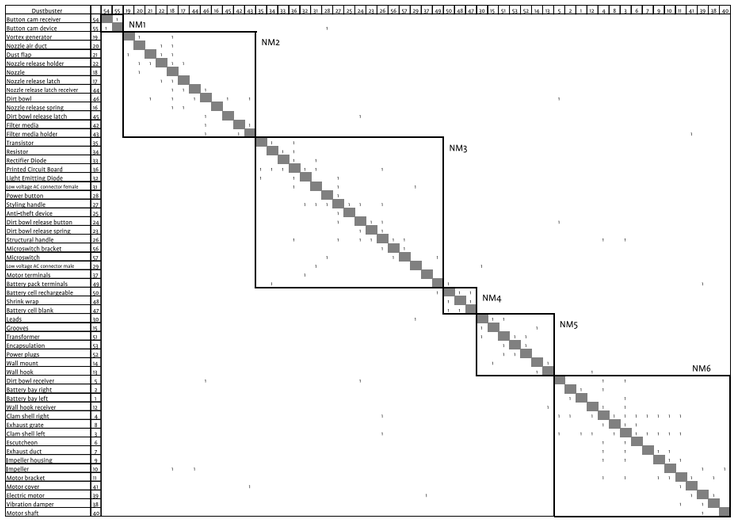
(Borjesson and Hölttä-Otto, 2012)			Proposed method		
Module	**No.**	**Component**	**Module**	**No.**	**Component**
M1	1	Battery bay left	NM1	54	Button cam receiver
	2	Battery bay right		55	Button cam device
M2	3	Clam shell left	NM2	19	Vortex generator
	4	Clam shell right		20	Nozzle air duct
	5	Dirt bowl receiver		21	Dust flap
	6	Escutcheon		22	Nozzle release holder
	7	Exhaust duct		18	Nozzle
	8	Exhaust grate		17	Nozzle release latch
	9	Impeller housing		44	Nozzle release latch receiver
	10	Impeller		46	Dirt bowl
	11	Motor bracket		16	Nozzle release spring
	12	Wall hook receiver		45	Dirt bowl release latch
M3	13	Wall hook		42	Filter media
	14	Wall mount		43	Filter media holder
	15	Grooves	NM3	35	Transistor
M4	16	Nozzle release spring		34	Resistor
	17	Nozzle release latch		33	Rectifier Diode
	18	Nozzle		36	Printed Circuit Board
	19	Vortex generator		32	Light Emitting Diode
	20	Nozzle air duct		31	Low voltage AC connector female
	21	Dust flap		28	Power button
	22	Nozzle release holder		27	Styling handle
M5	23	Dirt bowl release spring		25	Anti-theft device
	24	Dirt bowl release button		24	Dirt bowl release button
	25	Anti-theft device		23	Dirt bowl release spring
	26	Structural handle		26	Structural handle
	27	Styling handle		56	Microswitch bracket
	28	Power button		57	Microswitch
M6	29	Low voltage AC connector male		29	Low voltage AC connector male
	30	Leads		37	Motor terminals
	31	Low voltage AC connector female		49	Battery pack terminals
M7	32	Light Emitting Diode	NM4	50	Battery cell rechargeable
	33	Rectifier Diode		48	Shrink wrap
	34	Resistor		47	Battery cell blank
	35	Transistor	NM5	30	Leads
	36	Printed Circuit Board		15	Grooves
M8	37	Motor terminals		51	Transformer
	38	Vibration damper		53	Encapsulation
	39	Electric motor		52	Power plugs
	40	Motor shaft		14	Wall mount
M9	41	Motor cover		13	Wall hook
	42	Filter media	NM6	5	Dirt bowl receiver
	43	Filter media holder		2	Battery bay right
M10	44	Nozzle release latch receiver		1	Battery bay left
	45	Dirt bowl release latch		12	Wall hook receiver
	46	Dirt bowl		4	Clam shell right
M11	47	Battery cell blank		8	Exhaust grate
	48	Shrink wrap		3	Clam shell left
	49	Battery pack terminals		6	Escutcheon
	50	Battery cell rechargeable		7	Exhaust duct
M12	51	Transformer		9	Impeller housing
	52	Power plugs		10	Impeller
	53	Encapsulation		11	Motor bracket
M13	54	Button cam receiver		41	Motor cover
	55	Button cam device		39	Electric motor
M14	56	Microswitch bracket		38	Vibration damper
	57	Microswitch		40	Motor shaft

TABLE 1. *The resulting modules and components of the Dustbuster*

FIGURE 5. *The clustered DSMs and the resulting modules of the Dustbuster*



A. *The clustered DSM in Ref. (Borjesson and Hölttä-Otto, 2012)*



B. *The clustered DSM obtained using the proposed method*

authors

Sangjin Jung is a postdoctoral research associate at the Pennsylvania State University. He received his PhD and MS in Mechanical Engineering from Hanyang University in 2012 and 2007. He was a senior research engineer at LG Production Engineering Research Institute (PRI) in Modular Design Group. He is currently developing modularity assessment and product architecting methods. He is also interested in product family redesign, multidisciplinary design optimization, and value-driven design.

Timothy W. Simpson is a Professor of Mechanical and Industrial Engineering at Penn State with affiliate appointments in Engineering Design and the College of Information Sciences and Technology. His research interests include product family and product platform design, trade space exploration, multidisciplinary design optimization, and additive manufacturing. He has co-authored over 250 peer-reviewed journal and conference papers and served as the lead editor on two books on product family and product platform design. He is the recipient of the 2013 ASME Ben C. Sparks Award, the 2011 ASEE Fred Merryfield Design Award, and numerous awards for outstanding teaching and research at Penn State. He is a Fellow in ASME and an Associate Fellow in AIAA. He received his Ph.D. and M.S. degrees in Mechanical Engineering from Georgia Tech, and his B.S. in Mechanical Engineering from Cornell University.

references

Bäck, T. *(1996).* Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press.

Borjesson, F., Hölttä-Otto, K. *(2012).* Improved clustering algorithm for design structure matrix, ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, pp. 921-930.

Borjesson, F., Hölttä-Otto, K. *(2014).* A Module Generation Algorithm for Product Architecture based on Component Interactions and Strategic Drivers. Research in Engineering Design 25, 31-51.

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

Eppinger, S.D., Browning, T.R. *(2012).* Design Structure Matrix Methods and Applications. MIT Press, Cambridge, MA.

Fernandez, C.I.G. *(1998).* Integration analysis of product architecture to support effective team co-location. ME thesis, MIT, Cambridge, MA.

Gen, M., Cheng, R. *(1997).* Genetic algorithms and engineering design. Wiley.

Gershenson, J.K., Prasad, G.J., Zhang, Y. *(2004).* Product modularity: measures and design methods. Journal of Engineering Design 15, 33-51.

Grefenstette, J. *(1987).* Incorporating problem specific knowledge into genetic algorithms, in: Davis, L. *(Ed.)*, Genetic algorithms and simulated annealing. Morgan Kaufmann Publishers, Los Altos, CA.

Guo, F., Gershenson, J.K. *(2003).* Comparison of Modular Measurement Methods Based on Consistency Analysis and Sensitivity Analysis, in: Schmidt, L. *(Ed.)*, ASME Design Engineering Technical Conferences - Design Theory & Methodology. ASME, Chicago, IL.

Guo, F., Gershenson, J.K. *(2004).* A comparison of modular product design methods based on improvement and iteration, ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers, pp. 261-269.

Helmer, R., Yassine, A., Meier, C. *(2010).* Systematic module and interface definition using component design structure matrix. Journal of Engineering Design 21, 647-675.

Hölttä-Otto, K., de Weck, O. *(2007).* Degree of Modularity in Engineering Systems and Products with Technical and Business Constraints. Concurrent Engineering: Research and Applications 15, 113-126.

Jung, S., Park, G.-B., Choi, D.-H. *(2013).* A Decomposition Method for Exploiting Parallel Computing including Determination of an Optimal Number of Subsystems. ASME Journal of Mechanical Design 135, 041005 *(041009 pgs)*.

Kamrani, A.K., Gonzalez, R. *(2003).* A genetic algorithm-based solution methodology for modular design. Journal of Intelligent Manufacturing 14, 599-616.

Otto, K., Hölttä-Otto, K., Simpson, T.W. *(2013).* Linking 10 years of modular design research: alternative methods and tool chain sequences to support product platform design, ASME Design Engineering Technical Conferences, Portland, OR.

Rissanen, J. *(1978).* Modeling by shortest data description. Automatica 14, 465-471.

Rissanen, J. *(1999).* Hypothesis selection and testing by the MDL principle. The Computer Journal 42, 260-269.

Simpson, T.W. *(2004).* Product platform design and customization: status and promise. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18, 3-20.

Steward, D.V. *(1981).* Systems Analysis and Management: Structure, Strategy and Design. Petrocelli Books, Inc., New York.

Thebeau, R.E. *(2001).* Knowledge management of system interfaces and interactions for product development process, System design & management program. Massachusetts Institute of Technology, Cambridge, MA.

van Beek, T.J., Erden, M.S., Tomiyama, T. *(2010).* Modular design of mechatronic systems with function modeling. Mechatronics 20, 850-863.

Whitfield, R.I., Smith, J.S., Duffy, A.B. *(2002).* Identifying component modules, Artificial Intelligence in Design'02. Springer, pp. 571-592.

Yu, T.-L., Yassine, A.A., 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.

Yu, T.-L., Yassine, A.A., Goldberg, D.E. *(2007).* An Information Theoretic Method for Developing Modular Architectures Using Genetic Algorithms. Research in Engineering Design 18, 91-109.