Achille Messac
Associate Professor
Rensselaer Polytechnic Institute
Mechanical, Aerospace, and Nuclear Engineering Department
Multidisciplinary Design and Optimization Laboratory
110 8th Street
Troy, NY 12180
e-mail: messac@rpi.edu

Michael P. Martinez
Graduate Student
Northeastern University
Mechanical Engineering Department
Multidisciplinary Design Laboratory
360 Huntington Avenue
Boston, MA 02115

Timothy W. Simpson
Assistant Professor
The Pennsylvania State University
Departments of Mechanical and Nuclear Engineering and
Industrial & Manufacturing Engineering
329 Leonhard Building
University Park, PA 16802

Introduction of a Product Family Penalty Function Using Physical Programming

In an effort to increase customization for today's highly competitive global markets, many companies are looking to product families to increase product variety and shorten product lead-times while reducing costs. The key to a successful product family is the common product platform around which the product family is derived. Building on our previous work in product family design, we introduce a product family penalty function (PFPF) in this paper to aid in the selection of common and scaling parameters for families of products derived from scalable product platforms. The implementation of the PFPF utilizes the powerful physical programming paradigm to formulate the problem in terms of physically meaningful parameters. To demonstrate the proposed approach, a family of electric motors is developed and compared against previous results. We find that the PFPF enables us to properly balance commonality and performance within the product family through the judicious selection of the common parameters that constitute the product platform and the scaling parameters used to instantiate the product family.
[DOI: 10.1115/1.1467602]

1 Frame of Reference: Scalable Product Platforms

In an effort to increase customization for today's highly competitive global markets, many companies are looking to product families to increase product variety and shorten product lead-times while reducing costs. For example, Kodak's product platform-based response to Fuji's introduction of the QuickSnap single-use camera in 1987 enabled them to develop products faster and more cheaply through effective product family design, allowing them to regain market share and leapfrog Fuji [1]. Similarly, Airbus is currently enjoying a competitive advantage over Boeing due to improved commonality, particularly in the cockpit. The A330 cockpit is common to all other Airbus types while Boeing's 767-400 cockpit is common only with the 757. This enabled the A330-200, a less efficient ''shrink'' of a larger aircraft, to outsell Boeing's 767-400ER, a more efficient ''stretch'' design of a smaller aircraft, last year [2].

As evidenced by these and similar examples in the literature, the key to a successful product family is the common product platform around which the product family is derived. As Robertson and Ulrich [3] point out, ''By sharing components and production processes across a platform of products, companies can develop differentiated products efficiently, increase the flexibility and responsiveness of their manufacturing processes, and take market share away from competitors that develop only one product at a time.'' An effective product platform also facilitates customization by enabling a variety of products to be quickly and easily developed to satisfy the needs and requirements of distinct market niches [4]. However, product platform design requires carefully balancing the commonality of the product platform with the distinctiveness of the individual products that constitute the family.

The prominent approach to product platform and product family design is the development of modular product platforms that can be easily modified through the addition, substitution, and removal of one or more functional modules [5,6]. An alternative, and equally important, approach to product family design is the development of common product platforms that can be ''stretched'' or ''scaled'' in one more dimensions to satisfy a variety of market requirements [7]. This approach is frequently employed in aircraft design, whereby an aircraft such as the 777-X is stretched to accommodate more passengers, increase its flight range, or carry additional cargo [8].

To facilitate the design of scalable product platforms, we have been developing methods to synthesize common product platforms that can be scaled into an appropriate family of products. This work began by employing robust design principles to find common product platforms that could be scaled by one or more scale factors, e.g., the number of passengers on an aircraft [9,10]. This approach was then formalized into the Product Platform Concept Exploration Method (PPCEM), which employed a two-stage approach to find a common platform and product family based on a known scale factor [7]. Recently, a single-stage approach employing the powerful physical programming paradigm was developed to reduce computational expense and improve the performance of the product family [11].

In both approaches, the scale factor around which the product family was ''stretched'' was assumed to be known *a priori*; however, this is often not true in practice. In fact, there is a tradeoff between commonality and performance within a product family: improving commonality within a product family reduces the distinctiveness of the products. Selecting the right combination of common and scaling parameters is critical to the success of the product platform and corresponding family of products. Consequently, in this paper *we introduce a Product Family Penalty Function (PFPF) to aid in the selection of common and scaling parameters for families of products derived from scalable product platforms.*

The paper is organized as follows. Section 2 consists of a brief description of the PPCEM and a synopsis of the physical pro-

Contributed by the Design Automation Committee for publication in the JOURNAL OF MECHANICAL DESIGN. Manuscript received Aug. 2000. Associate Editor: J. E. Renaud.

gramming method. The PFPF is described and developed in Section 3. An example problem is presented in Section 4. Section 5 provides concluding remarks.

2 Technological Basis

2.1 Product Platform Concept Exploration Method (PPCM) [7]. The PPCEM consists of five steps that prescribe how to formulate the product family problem and describe how to solve it; the actual implementation of each step is likely to vary from problem to problem. A quick overview of each step in the process follows.

Step 1: Create the Market Segmentation Grid—This step involves mapping the overall design requirements into an appropriate market segmentation grid. The grid allows for identification of potential opportunities for leveraging the product platform.

Step 2: Classify the Factors and Ranges—This step involves mapping the overall design requirements and market segmentation grid into appropriate factors and identifying corresponding ranges for each scale factor. Scale factors should be identified in this step based on the platform leveraging strategies identified in Step 1.

Step 3: Build and Validate Metamodels—This step includes modeling computationally expensive computer analyses or simulation codes using computationally inexpensive metamodels (e.g., response surfaces, kriging).

Step 4: Aggregate Product Platform Specifications—This step includes formulating an optimization problem based on information from the market segmentation grid, the factors and ranges, and the overall design requirements. Physical programming is used to formulate this problem.

Step 5: Develop the Product Platform Portfolio—Use the optimization formulation from Step 4 to obtain the product platform portfolio that optimizes the design objectives.

By using the PPCEM, a strategy for developing a product platform portfolio based on a single platform is readily available; however, depending on the optimization formulation, practically any combination of solutions is possible, including both good and bad designs. Even if the optimization problem is formulated carefully and correctly, bad designs are possible, even likely. A key reason these potentially bad designs are the poor choices in determining common and scaling parameters.

Steps 1 and 2 of the PPCEM utilize market segmentation grids and platform leveraging strategies, but they do not clearly address the issue of how to choose common and scaling parameters within the product family. Due to the nonlinear nature of multi-criteria optimization, the task of choosing the scaling parameters cannot rely on intuition. After the physical programming overview in the next section, we introduce a product family penalty function based on the physical programming paradigm to aid in the determination of common and scaling parameters within a product family.

2.2 Physical Programming Synopsis. This section provides a synopsis of the physical programming (PP) method. For a comprehensive presentation, see Ref. [12]. Physical programming is intended to be a simple and user-friendly optimization method that requires negligible knowledge of optimization. The application of physical programming employs a flexible and natural problem formulation framework. In PP, the designer does not need to specify optimization weights in the problem formulation phase. Rather, the designer specifies ranges of different degrees of desirability for each design objective.

Physical programming also addresses the inherent multiobjective nature of design problems, where multiple conflicting objectives govern the search for the *best* solution. Physical programming provides a flexible and more deterministic approach to obtaining a solution that satisfies the typically complex texture of a designer's preferences. Physical programming is implemented in the software package entitled PhysPro [13], which is Matlab

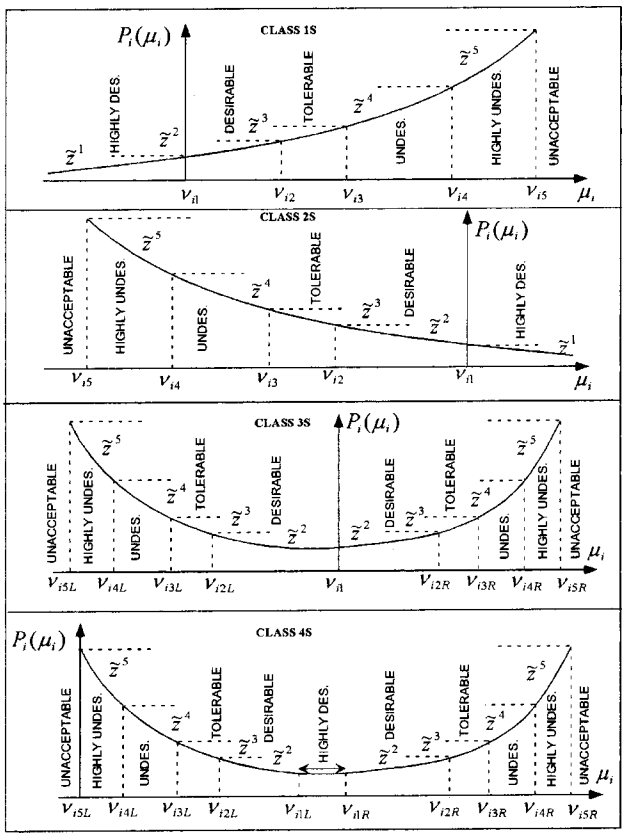

Fig. 1 Class-function ranges for *i*th generic design metric

based. Some of the important concepts used in the PP method are described as follows. More information on the physical programming method can be found in [14–21].

Design Metrics. The problem formulation involves identifying the characteristics of the system, or design, which allow the designer to judge the effectiveness of the outcome. Those characteristics, or design metrics, are denoted by μ_i, which are components of the vector $\mu=(\mu_1 \ldots, \mu_m)$. The elements μ_i represent behavior. Design metrics may be quantities that the designer wishes to minimize; maximize; take on a certain value (goal); fall in a particular range; or be less than, greater than, or equal to particular values. The designer defines preference with respect to each design metric by providing certain numerical values. The design metric may become part of an *aggregate objective function* (AOF) that will be minimized, or may instead be treated as inequality or equality constraints that subjugate the aggregate objective function.

Classification of Objectives, and Class Functions. Within the physical programming procedure, the engineer expresses objectives with respect to each design metric using four different *Classes*. Each *Class* comprises two cases, *Hard* and *Soft*, referring to the sharpness of the preference. All *Soft* Class functions will become constituent components of the aggregate objective function. Figure 1 depicts the qualitative meaning of each *Soft Class*. The value of the design metric under consideration, μ_i, is on the horizontal axis, and the function that will be minimized for that objective, $P_i(\mu_i)$, hereby called the *Class function*, is on the vertical axis.

The desired behavior of a generic design metric is described by one of eight sub-Classes: four Soft and four Hard. These Classes are characterized by:

Soft:

Class 1S Smaller-Is-Better, i.e., minimization.

Class 2S Larger-Is-Better, i.e., maximization.
Class 3S Value-Is-Better.
Class 4S Range-Is-Better.
Hard:
Class 1H Must be smaller, i.e., $\mu_i \leqslant \mu_{i,\max}$
Class 2H Must be larger, i.e., $\mu_i \geqslant \mu_{i,\min}$
Class 3H Must be equal, i.e., $\mu_i = \mu_{i,val}$
Class 4H Must be in range, i.e., $\mu_{i,\min} \leqslant \mu_i \leqslant \mu_{i,\max}$.

For each of these Classes, we form a *Class function* (Fig. 1). The Class functions provide the means for a designer to express the spectrum of preferences for a given design metric. The Class functions provide information that is deliberately imprecise. By design, the utopian value of each Class functions is zero. Next, we explain how quantitative specifications are associated with each design metric.

Physical Programming Lexicon. Physical programming defines a lexicon that provides the means to express preference in a flexible way. This lexicon comprises terms that characterize the degree of desirability of six ranges for each generic design metric for Classes 1S and 2S, ten ranges for Class 3S, and eleven for Class 4S. To illustrate, consider the case of Class 1S. The ranges are defined as follows, in order of decreasing preference:

Highly Desirable range ($\mu_i \leqslant \nu_{i1}$): An acceptable range over which the improvement that results from further reduction of the performance objective is desired, but is of minimal additional value.

Desirable range ($\nu_{i1} \leqslant \mu_i \leqslant \nu_{i2}$): An acceptable range that is desirable.

Tolerable range ($\nu_{i2} \leqslant \mu_i \leqslant \nu_{i3}$): An acceptable, tolerable range.

Undesirable range ($\nu_{i3} \leqslant \mu_i \leqslant \nu_{i4}$): A range that, while acceptable, is undesirable.

Highly Undesirable range ($\nu_{i4} \leqslant \mu_i \leqslant \nu_{i5}$): A range that, while still acceptable, is highly undesirable.

Unacceptable range ($\mu_i \geqslant \nu_{i5}$): The range of values that the generic objective may not take.

The parameters ν_{i1} through ν_{i5} are physically meaningful constants that are specified by the designer to quantify the preference associated with the *i*th design metric. These parameters delineate the ranges for each design metric.

The Class functions map design metrics into non-dimensional, strictly positive real numbers. This mapping, in effect, transforms design metrics with disparate units and physical meaning onto a dimensionless scale through a unimodal function. Figure 1 illustrates the mathematical nature of the Class functions and shows how they allow a designer to express the ranges of differing goodness, or preferences, for a given design metric. Consider the first curve of Fig. 1: the Class function for Class 1S design metrics. Six ranges are defined. The parameters ν_{i1} through ν_{i5} are specified by the designer. When the value of the objective μ_i is less than ν_{i1} (highly-desirable range), the value of the Class function is small; which calls for little further minimization of the Class function. When, on the other hand, the value of the objective μ_i is between ν_{i4} and ν_{i5} (highly-undesirable range), the value of the Class function is large; which calls for significant minimization of the Class function. The behavior of the other Soft Class functions is indicated in Fig. 1. Preferences regarding each design metric are treated independently, allowing the inherent multiobjective nature of the problem to be preserved. This describes the basic process of physical programming: the value of the Class function for each design metric governs the optimization path in objective space.

3 Product Family Penalty Function Development

We propose a Product Family Penalty Function (PFPF) that will optimize the design of a product family. The PFPF will determine which parameters should be common throughout the product family and which should be the scaling variables. Here, we assume the use of a single scaling variable within the product family; however, the method can easily be extended to include searching for multiple scaling parameters.

During optimization, the PFPF will penalize design parameters that are not common throughout the product family while optimizing the desired objectives. This will allow for identification of the design parameters that are best declared common and those to scale the common product platform. If the design parameters can be set at a constant value for all products, with minimal effect on the design objectives, then they should be grouped into the common product platform. If, however, a parameter cannot be made constant across the products without adversely affecting the design objectives, then it should be considered a good candidate for becoming a scaling parameter.

The methodology that is being examined is to first optimize each product individually. This gives two important results: (1) an optimum configuration for each product, for later comparison, and (2) initial conditions for the next optimization problem, including the minimization of the variation of the design parameters. With the starting points obtained (optimum individual configurations), the second optimization problem can be solved. The desired design objectives are optimized, while in addition, the variations between the same design parameters across all the products in the family are also minimized. As an example, suppose there are *k* products in a family. The *variation* for the *i*th design parameter is defined as

$$\mathrm{var}_i = \sqrt{\sum_{j=1}^{k} \frac{(x_i^j - \overline{x_i})^2}{k}}$$

where

$$\overline{x_i} = \sum_{j=1}^{k} \frac{x_i^j}{k}$$

and x_i^j denotes the *i*-th design parameter for the *j*-th product. The variation will always be considered as a percentage of the mean for the variable being considered. That is, as the variables change during the optimization, the percentage variation is based on the correct mean of the variables. For this example, we minimize the variation of the design parameters. This results in a set of design metrics, in addition to the original existing metrics.

In order to demonstrate that physical programming has the ability to handle the optimization of numerous objectives and give reliable results, the first part of the PFPF, finding the variable best suited for the scaling variable, will be performed using two methods.

The first method will be more time consuming, where the variation of each design parameter will be minimized across the family, thus the number of optimization runs will be equal to the number of design parameters that are being considered for scaling parameters. Minimizing the variation of each design parameter, while letting all others be unconstrained, will in effect allow the designer to see how much performance is lost when forcing the variation to go to zero (becoming a variable that is the same, or common, across all products in the family). When this is performed for all design parameters, one can see how each parameter responds to being forced common. The design variable that causes the largest decrease in performance is deemed best suited to be the scaling variable.

The second method, which will be more efficient and also take into consideration all design objectives and constraints in the AOF at the same time, will be based on a single formulation using the physical programming method. This method will optimize each performance design objective while also minimizing the variations of all the design parameters considered for scaling variables. It will later be explained how this can be accomplished in the physical programming paradigm. Next a realistic problem is considered where we optimize a family of 10 universal motors.

4 Example Problem: Family of Universal Motors

4.1 Universal Motors. Universal electric motors are so named for their capability to function on both direct current (DC) and alternating current (AC). Universal motors deliver more torque for a given current than any single-phase motor [22]. The high performance characteristics and flexibility of universal motors have led to a wide range of applications, especially in household use where they are found in, e.g., electric drills and saws, blenders, vacuum cleaners, and sewing machines [23].

According to Lehnerd [24], in the 1970s Black & Decker developed a family of universal motors for its power tools in response to a new safety regulation: double insulation. Prior to that, they used different motors in each of their 122 basic tools with hundreds of variations, from jigsaws and grinders to edgers and hedge trimmers. Through redesign and standardization of the product line, they were able to *produce all their power tools using a line of motors that varied only in the stack length* and the amount of copper wrapped within the motor. As a result, all of the motors could be produced on a single machine with stack lengths varying from 0.8 in to 1.75 in, and power output ranging from 60 to 650 watts. In addition to significant material and labor savings, new designs were developed using standardized components such as the redesigned motor, allowing products to be introduced, exploited and retired with minimal expense related to product development.

Our goal in this example is to demonstrate the use of the PFPF in conjunction with the PPCEM to design a family of universal motors. Specifically, the objective is to:

Design a family of ten (10) universal electric motors that satisfy a variety of torque and power requirements, built on one or more common platforms, and scaled around a set of scaling parameters.

The motor platform is the set of common physical dimensions (design variables) that describe the universal motor while the scale factor is varied to satisfy the range of torque and power requirements.

A schematic of a universal motor is shown in Fig. 2. As shown in the figure, a universal motor is composed of an armature and a field, which are also referred to as the rotor and stator, respectively. The armature consists of a metal shaft and slats (armature poles) around which wire is wrapped longitudinally as many as a thousand times. The field consists of a hollow metal cylinder within which the armature rotates. The field also has wire wrapped longitudinally around interior metal slats (field poles) as many as hundreds of times. For a universal motor, the wire wrapped around the armature and the field is wired in series, which means that the same current is applied to both set of wires.

As current passes through the field windings, a large magnetic field is generated. This field passes through the metal of the field, across an air gap between the field and the armature, then through the armature windings, through the shaft of the armature, across another air gap, and back into the metal of the field, thus completing a magnetic circuit. When the magnetic field passes though the armature windings, the magnetic field exerts a force on the current carrying wires. Because of the geometry of the windings, current on one side of the armature is always passing in the opposite direction to the current on the other side of the armature. Thus the force exerted by the magnetic field on one side of the armature is opposite to the force exerted on the other side of the armature. Thereby a net torque is exerted on the armature, causing the armature to spin.

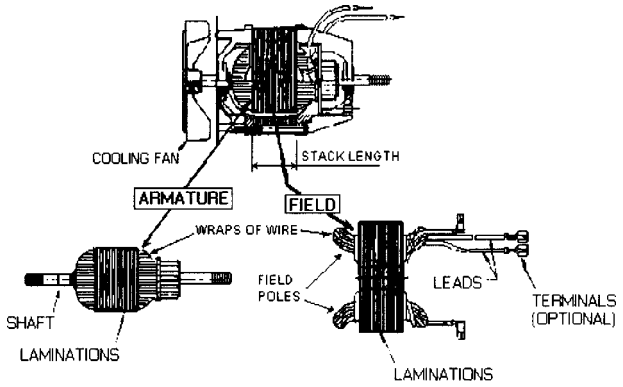

Fig. 2 Universal Motor Schematic (G.S. Electric [25])

4.2 Example Problem. An example using ten universal motors is now considered. There are two separate problems that must be solved. The first optimization problem involves the minimization of the mass (kg) and the maximization of the efficiency (%) for each motor. There are eight design parameters for each motor. The design parameters are as follows: the current drawn by the motor (I); the wires' cross-sectional areas (A_{wf}, A_{wa}) and the numbers of turns (N_s, N_c) in both the field and the armature; and the radius (r_o), thickness (t), and stack length (L) of the motor. The constraints are as follows:

Torque,

$$T=\left\{\begin{array}{l}0.05,0.10,0.125,0.15,0.20,\\ 0.25,0.30,0.35,0.40,0.50\end{array}\right\}$$

Nm for each of the ten motors, respectively,

Power,

$$P=300 \text{ watts for all ten motors,}$$

Feasible Geometry,

$$\frac{r_o}{t}\geqslant 1,$$

and Magnetizing Intensity,

$$H\leqslant 5000 \text{ Amp*turns/m.}$$

Using the formulation derived in Simpson et al. [7], from Chapman [22] and Cogdell [26], a physical programming optimization formulation was developed. The preferences for the formulation can be seen in Table 1. The results of the individual optimization runs for each motor can be seen in Table 2. The results from this first optimization problem will be used in the formulation of the second optimization problem for starting points. The individually optimized motors will also be used as a baseline to compare later results.

Multiple Formulation Method. The first method for determining which design parameter is best suited for the scaling variable will now be demonstrated. This method is implemented by performing seven separate optimization runs since the eighth variable current, I, is not considered a candidate for scaling since it is a state variable. This will be done by forcing the variation of each of the seven candidates for scaling variable to be zero throughout the family while letting all other variables be free. The preference structure for the first optimization run can be seen in Table 3 (all the preferences from Table 1 are also used). Note that the first variation metric in Table 3 is a Class 3-H, or equality constraint, while the other variation metrics are Class 2-H, or greater than constraints. For this problem we took a representative sample of the motors that were in the family. In other words, we used motors 1, 5, and 10 to note the trend in performance that was lost when commonality was forced upon each design parameter.

In all seven optimization runs, a performance loss was noted for mass and efficiency in each of the runs. The mass losses when each variation metric was minimized can be seen in Fig. 3 while the efficiency losses can be seen in Fig. 4. Now both of these performance losses must be combined into one graph to allow the

designer to use this information to make the best decisions regarding which variable should become the scaling variable. Combining these two sets of results into one graph could be simply done by adding the percentages lost or gained from mass and efficiency additively. However, this would not allow the results to be kept in a truly physically meaningful environment. Meaning that if one objective was more important or weighted more strongly than the other, this method would not capture and reflect this physical meaning. For this case, a method based on the preferences that were previously entered to describe the designer's physically meaningful choices will be used. This will allow for the physically meaningful information from the preferences to be captured in combining the two objectives.

Since we looked at a representative sample of the three motors (1, 5, and 10) in this case, we will refer to the preferences entered in Table 1. A simple method was developed to map these two disparate objectives to the same graph. Keeping in mind that the scaling variable should be the design variable with the most performance lost, a mapping is created. Since there are two objectives, and the total of the aggregate objective function (AOF)

Table 1 Physical programming preferences for example problem

		HD ↔	D ↔	T ↔	U ↔	HU ↔	
*i*th Objective	Class	g_{i1}	g_{i2}	g_{i3}	g_{i4}	g_{i5}	
Mass - 1	1-S	.20	.30	.40	.50	.60	
Efficiency - 1	2-S	.85	.80	.75	.70	.65	
Mass - 2	1-S	.25	.35	.45	.55	.65	
Efficiency - 2	2-S	.80	.75	.70	.65	.60	
Mass - 3	1-S	.30	.40	.50	.60	.70	
Efficiency - 3	2-S	.80	.75	.70	.65	.60	
Mass - 4	1-S	.30	.40	.50	.60	.70	
Efficiency - 4	2-S	.80	.75	.70	.65	.60	
Mass - 5	1-S	.30	.40	.50	.60	.70	
Efficiency - 5	2-S	.75	.70	.65	.60	.55	
Mass - 6	1-S	.35	.45	.55	.65	.75	
Efficiency - 6	2-S	.75	.70	.65	.60	.55	
Mass - 7	1-S	.45	.55	.65	.75	.85	
Efficiency - 7	2-S	.75	.70	.65	.60	.55	
Mass - 8	1-S	.45	.55	.65	.75	.85	
Efficiency - 8	2-S	.70	.65	.60	.55	.50	
Mass - 9	1-S	.55	.65	.75	.85	.95	
Efficiency - 9	2-S	.65	.60	.55	.50	.45	
Mass - 10	1-S	.60	.70	.80	.90	1.0	
Efficiency - 10	2-S	.60	.55	.50	.45	.40	
			Unacceptable				Acceptable
Mag Int. (1-10)	1-H	-	-	-	-	5000	
Feasibility (1-10)	2-H	-	-	-	-	1	
Power (1-10)	3-H	-	-	-	-	300	

HU: *Highly Undesirable*, U: *Undesirable*, T: *Tolerable*, D: *Desirable*, HD: *Highly Desirable*.

Table 2 Individually optimized motors using physical programming

Motor	I (Amp)	L (cm)	N_c (turns)	N_s (turns)	R_o (cm)	t (cm)	A_{wf} (mm^2)	A_{wa} (mm^2)	Eff. (%)	Mass (kg)
1	2.97	1.90	691	70	1.70	.424	.241	.241	85.1	0.275
2	3.48	1.97	907	69	1.91	.492	.216	.216	75.1	0.358
3	3.48	2.11	942	74	2.04	.542	.239	.239	75.0	0.439
4	3.57	2.16	1000	75	2.13	.579	.246	.246	73.1	0.490
5	4.01	2.26	1059	69	2.20	.581	.228	.228	65.0	0.519
6	4.01	2.33	1146	74	2.35	.664	.256	.256	65.0	0.636
7	4.06	2.45	1184	76	2.47	.712	.274	.274	64.3	0.739
8	4.50	2.46	1217	70	2.50	.712	.263	.263	58.0	0.740
9	4.74	2.50	1248	68	2.55	.736	.268	.268	55.0	0.784
10	5.22	2.55	1299	64	2.66	.784	.279	.279	50.0	0.867

Table 3 Physical programming preferences for example problem

		HD ↔	D ↔	T ↔	U ↔	HU ↔	
*i*th Objective	Class	g_{i1}	g_{i2}	g_{i3} Unacceptable	g_{i4}	g_{i5}	Acceptable
Variation of L	3-H						0
Variation of N_c	2-H						0
Variation of N_s	2-H						0
Variation of r_o	2-H						0
Variation of t	2-H						0
Variation of A_{wf}	2-H						0
Variation of A_{wa}	2-H						0

HU: *Highly Undesirable*, U: *Undesirable*, T: *Tolerable*, D: *Desirable*, HD: *Highly Desirable*.

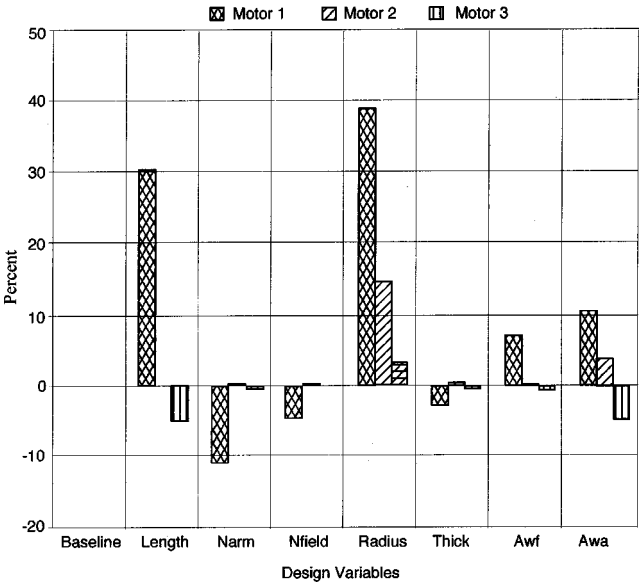

Fig. 3 Percent mass - 3 Motors

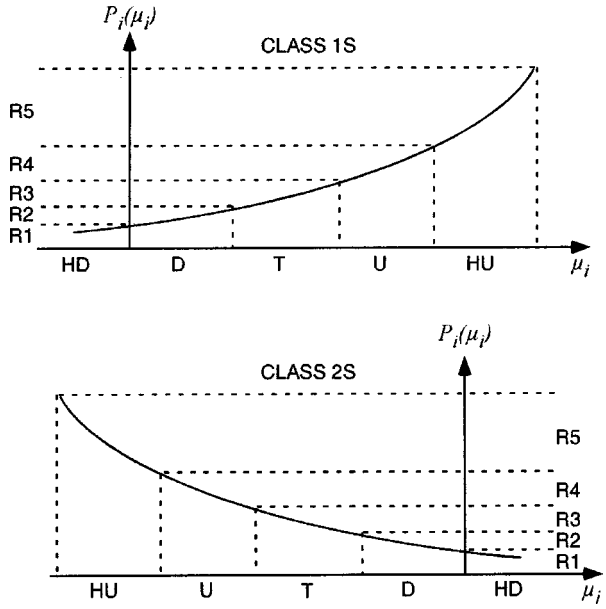

Fig. 5 Mapping the preferences

should equal one, each objective will have a maximum mapped value of 0.5. Figure 5 shows two class functions showing how the mapping takes place, where R1 through R5 represent the five preference ranges. Table 4 summarizes the range values and gives their mapped values.

Using this mapping, the results from mass and efficiency can be combined into one graph. This is now done in a physically meaning way. If by forcing the variation of a design variable to be zero, the mass and efficiency objectives end up as two highly undesirable results then the mapping would cause the combination of the two to total one, or the highest possible total. Two highly desirable results would bring the total to near zero. The higher the total, the better suited the variable is to become the scaling variable since it has a large negative impact on the performance of the family if it is taken as common. Figure 6 allows the designer to see the combination of the two objectives after they have been mapped. A clearer representation of this is seen in Fig. 7, where the lowest value for each motor is subtracted to allow the difference between the design parameters to be seen.

It is seen from Fig. 7 that, when radius is forced to be common across all products, there is a large price in performance to be paid. Figure 7 shows that the radius might be the best choice for the scaling variable as opposed to the length, which was chosen for the problem by Black & Decker.

This provides a good tool for the designer to use; however, by employing this type of method all of the other design parameters are free to vary. Thus, there may be some interactions between these parameters and design objectives that are not being represented clearly. That is why it is desired to optimize everything at once and see how the system responds when all design objectives are in the AOF. This is what will be explained and shown in the next sub-section.

Table 4 Summary of range values

Preference	Range	Value
HU	R5	0.500
U	R4	0.375
T	R3	0.250
D	R2	0.125
HD	R1	0.000

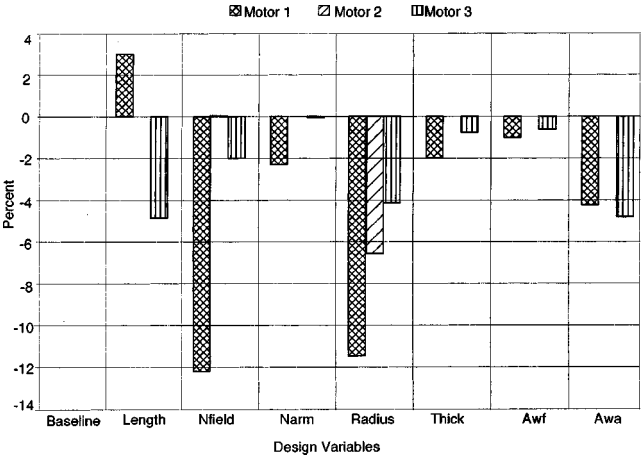

Fig. 4 Percent efficiency - 3 Motors

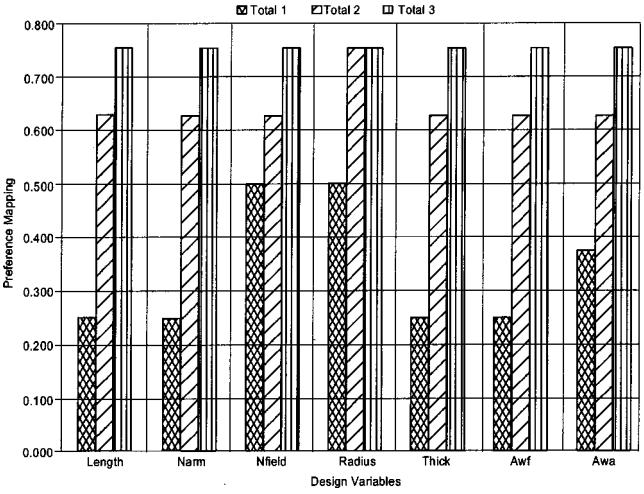

Fig. 6 Mass and efficiency combined

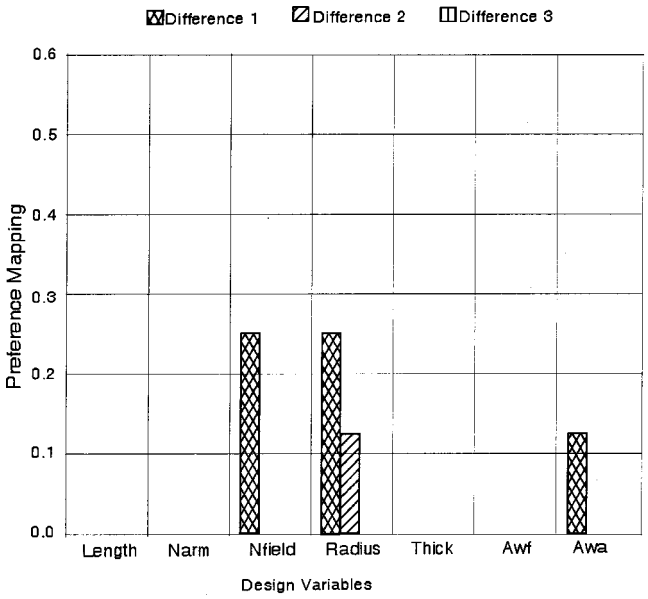

Fig. 7 Design tool for determining scaling parameter

Single Formulation Method. In formulating this problem as a single optimization, physical programming will be used in a slightly different way than has been discussed in previous papers. The physical programming method will be the same, the new twist that will be looked at will involve how the results are examined, and will be explained later on in this section.

Physical programming (PP) has the unique feature of forming its AOF based on physically meaningful preferences. For this example, the general formulation will be the same, the preferences from Table 1 will be used, with all of the variation design metrics also being minimized as seen in Table 5. These preferences (Table 5) are percentages of variations, so all variables are treated appropriately regardless of their magnitude. This formulation uses the preferences from Table 1 to minimize the mass of each motor, maximize the efficiency of each motor, and satisfy the given constraints. This formulation thereby *optimizes the performance* of the family of motors. The second set of preferences (in Table 5) minimizes the variation between each design variable throughout the family. Since both of these sets of preferences are acting against each other (by increasing commonality it decreases performance) the AOF can now use the optimization engine to determine which design parameter affects performance the most. This will cause PP to: *not minimize the variation of the design parameter that effects performance the most as much as PP will minimize the other design parameters variation*. Thus, with all of the factors considered in the AOF, by performing a single optimization run, it can easily be seen which factor would be best suited for the scaling variable.

To obtain a representative sample we will first examine two motors at a time: motors 1 and 5, then motors 5 and 10. We then use three and five motors at a time as the representative sample. Upon performing the optimization for motors 1 and 5, and 5 and 10, we can see the values that were produced for the variation design metrics in Table 6. Table 6 shows that the hardest variation design metric to minimize, while taking into consideration performance, is the radius. This result agrees with the previous result obtained using seven optimization runs. This leads us to conclude that radius would be the best suited for the scaling variable. As can be seen from Table 6, using 2, 3, or 5 motors yields the same conclusion; the radius was shown to be the best choice for a scaling variable. Radius had the highest percentage variation in each trial; this means it was the hardest variable to make common without sacrificing performance.

Table 6 Design metrics variance (%)

Design Metric	Motors 1,5	Motors 5,10	Motors 1,5,10	Motors 1,3,5,8,10
Var. of L	6.96	5.59	9.09	14.46
Var. of N_c	6.97	2.37	9.10	8.46
Var. of N_s	6.97	6.98	9.00	5.27
Var. of r_o	23.75	21.26	9.64	17.69
Var. of t	6.96	6.98	9.00	10.90
Var. of A_{wf}	4.49	1.51	8.98	13.34
Var. of A_{wa}	6.96	6.66	9.00	7.41

In this type of product family, since the product (universal motor) is being scaled to a more powerful version for each succeeding product, choosing representative motors give a sampling of the results. Ideally, all ten motors would be used at once, and there would be one optimization run needed, including all motors. However, the increasing number of conflicting objectives would cause the formulation to take an extremely long time to run, and could possibly generate many local minima. The average number of iterations when using two motors was approximately 4,250; three motors: 6,770; and five motors: 16,200. Note that the results when using 5 motors were found by taking initial conditions close to the optimal configurations for each motor. When different staring points were used, other local minimums were captured. This was not true for the case of using two motors. Any starting point resulted in the same solution.

Next, we will examine the results for the optimization of the universal motor product family. We will also examine the results from a previous study where physical programming was shown to be effective in product family optimization [11]. In this previous study, length was assumed to be the scaling variable, and the problem was solved using the Compromise DSP and the physical programming method. Now, the current physical programming solution that determined that the radius should be used as the scaling variable will be compared to these other solutions. This will allow us to see how changing the radius to be the scaling parameter affects the results. The final results using the PP method with radius as the scaling variable can be seen in Table 7. A comparison of these results to those from other methods is seen in the next section.

Table 5 Physical programming preferences for example problem

*i*th Objective	Class	HD ↔ g_{i1}	D ↔ g_{i2}	T ↔ g_{i3}	U ↔ g_{i4}	HU ↔ g_{i5}
Variation of L	1-S	1.5	4.5	7	9	10.5
Variation of N_c	1-S	1.5	4.5	7	9	10.5
Variation of N_s	1-S	1.5	4.5	7	9	10.5
Variation of r_o	1-S	1.5	4.5	7	9	10.5
Variation of t	1-S	1.5	4.5	7	9	10.5
Variation of A_{wf}	1-S	1.5	4.5	7	9	10.5
Variation of A_{wa}	1-S	1.5	4.5	7	9	10.5

HU: *Highly Undesirable*, U: *Undesirable*, T: *Tolerable*, D: *Desirable*, HD: *Highly Desirable*.

Comparison of Results. In making a comparison between the two product family designs using different scaling parameters, we first present, in Table 8, a comparison between the individually optimized product family and the product family design using the CDSP method employed in the original PPCEM formulation [7]. In Table 9, a comparison between the individually optimized product family and the physical programming formulation with length as the scaling variable can be seen. Finally in Table 10 we present the results of the current study, showing a comparison between the individually optimized product family and the physi-

Table 7 Results using PFPF for common platform development

Motor	I (Amp)	R_o (cm)	N_c (turns)	N_s (turns)	L (cm)	t (cm)	A_{wf} (mm^2)	A_{wa} (mm^2)	Eff. (%)	Mass (kg)
1	3.18	1.46	1319	68	2.12	.922	.256	.256	82.0	0.312
2	3.40	1.83	1319	68	2.12	.922	.256	.256	76.6	0.422
3	3.52	1.98	1319	68	2.12	.922	.256	.256	74.1	0.472
4	3.64	2.11	1319	68	2.12	.922	.256	.256	71.6	0.518
5	3.90	2.33	1319	68	2.12	.922	.256	.256	67.0	0.595
6	4.16	2.48	1319	68	2.12	.922	.256	.256	62.7	0.653
7	4.43	2.59	1319	68	2.12	.922	.256	.256	58.9	0.693
8	4.71	2.65	1319	68	2.12	.922	.256	.256	55.4	0.719
9	4.99	2.68	1319	68	2.12	.922	.256	.256	52.2	0.732
10	5.58	2.69	1319	68	2.12	.922	.256	.256	46.8	0.734

Table 8 Individual PP vs. CDSP

	Ind. PP		CDSP (Length Scaling)		Percentage Difference	
Motor	n (%)	m (kg)	n (%)	m (kg)	n (%)	m (kg)
1	85.1	0.275	76.8	0.380	−9.7	38.2
2	75.1	0.358	72.2	0.520	−3.8	45.4
3	75.0	0.439	70.0	0.576	−6.7	31.3
4	73.1	0.490	67.9	0.625	−7.2	27.4
5	65.0	0.519	63.9	0.703	−1.8	35.5
6	65.0	0.636	60.2	0.759	−7.4	19.3
7	64.3	0.739	56.8	0.797	−11.7	7.9
8	58.0	0.740	53.6	0.820	−7.6	10.9
9	55.0	0.784	50.5	0.830	−8.2	5.9
10	50.0	0.867	44.8	0.820	−10.4	−5.4
				Average Change	−7.4	21.6

Table 9 Individual PP vs. PP length scaling

	Ind. PP		PP (Length Scaling)		Percentage Difference	
Motor	n (%)	m (kg)	n (%)	m (kg)	n (%)	m (kg)
1	85.1	0.275	76.1	0.384	−10.5	39.7
2	75.1	0.358	71.9	0.506	−4.2	41.4
3	75.0	0.439	70.0	0.556	−6.7	26.9
4	73.1	0.490	68.1	0.601	−6.9	22.5
5	65.0	0.519	64.5	0.674	−0.8	29.8
6	65.0	0.636	61.2	0.729	−6.0	14.5
7	64.3	0.739	58.0	0.768	−9.8	3.9
8	58.0	0.740	55.0	0.768	−5.3	3.8
9	55.0	0.784	52.1	0.809	−5.3	3.2
10	50.0	0.867	46.7	0.811	−6.7	−6.5
				Average Change	−6.2	17.9

Table 10 Individual PP vs. PP radius scaling

	Ind. PP		PP (Radius Scaling)		Percentage Difference	
Motor	n (%)	m (kg)	n (%)	m (kg)	n (%)	m (kg)
1	85.1	0.275	82.0	0.312	−3.6	13.4
2	75.1	0.358	76.6	0.422	2.1	18.0
3	75.0	0.439	74.1	0.472	−1.3	7.6
4	73.1	0.490	71.6	0.518	−2.1	5.6
5	65.0	0.519	67.0	0.595	3.0	14.6
6	65.0	0.636	62.7	0.653	−3.5	2.6
7	64.3	0.739	58.9	0.693	−8.4	−6.2
8	58.0	0.740	55.4	0.719	−4.5	−2.8
9	55.0	0.784	52.2	0.732	−5.0	−6.6
10	50.0	0.867	46.8	0.734	−6.5	−15.3
				Average Change	−3.0	3.1

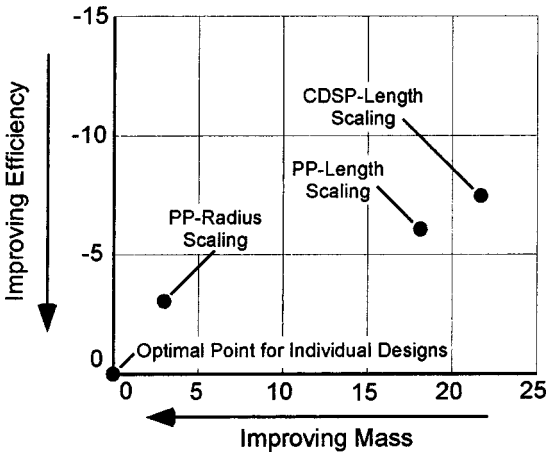

Fig. 8 Comparison of solutions

cal programming formulation with radius as the scaling variable. Figure 8 shows graphically how the three solutions compare to each other.

The lower-leftmost corner in Fig. 8 is the most desirable location, as mass is decreasing and efficiency is increasing. Keep in mind that this is only one of the possible Pareto solutions to the problem based on the given preferences. The two solutions where length was used as the scaling parameter can be seen on the right side of the graph. The CDSP Length Scaling solution gained approximately 21.6% in mass and lost approximately 7.4% in efficiency. The PP-Length Scaling solution gained approximately 17.9% in mass and lost approximately 6.2% in efficiency. This was slightly better than the CDSP method; however, by determining which parameter was best suited for the scaling parameter through the newly introduced PFPF optimization method, it can now be clearly seen that using radius as the scaling parameter provides significant improvement over using length. The PP-Radius Scaling solution yielded a gain of only 3.1% in mass and only a 3.0% decrease in efficiency. This results in a significant improvement when using radius as the scaling variable rather than the stack length.

5 Concluding Remarks

The main objective in this paper was to introduce a method to aid in the selection of common and scaling parameters for families of products derived from scalable product platforms to properly balance commonality and performance within a product family. In this study we found radius to be the best suited for the scaling variable as opposed to length, which has been used in previous studies. The PFPF is an important addition to the PPCEM, in particular, and product family design, in general. The art of uncovering which parameters should be common and which should be used for scaling is an important and difficult task. In product family design it is crucial to know which design variable should be the scaling parameter. Even if an efficient optimization method is used to find the optimally designed product family, if it is not based on using the best choice for a scaling parameter, then the truly optimal product family design cannot be achieved. Physical programming was able to effectively and efficiently find the proper scaling variable using a single optimization problem formulation. Thus, all design objectives could be considered in the AOF, allowing the optimization engine and the PP method to determine the best solution.

6 Acknowledgment

This research was partially supported by National Science Foundation Grant DMI–0196243.

References

[1] Wheelwright, S. C., and Clark, K. B., 1995, *Leading Product Development*, Free Press, New York.
[2] Aboulafia, R., 2000, "Airbus Pulls Closer to Boeing," Aerosp. Am., **38**(4), pp. 16–18.
[3] Robertson, D., and Ulrich, K., 1998, "Planning Product Platforms," Sloan Manage. Rev., **39**(4), pp. 19–31.
[4] Pine, B. J., II, 1993, *Mass Customization: The New Frontier in Business Competition*, Harvard Business School Press, Boston, MA.
[5] Pine, II, J. B., 1993, "Mass Customizing Products and Services," Planning Review, **22**(4), pp. 6(8).
[6] Ulrich, K., 1995, "The Role of Product Architecture in the Manufacturing Firm," Research Policy, **24**(3), pp. 419–440.
[7] Simpson, T. W., Maier, J. R. A., and Mistree, F., 2001, Product Platform Design: Method and Application, Research in Engineering Design, 13:1 (2-22).
[8] Sabbagh, K., 1996, *Twenty-First Century Jet: The Making and Marketing of the Boeing 777*, Scribner, NY.
[9] Simpson, T. W., Chen, W., Allen, J. K., and Mistree, F., 1996, "Conceptual Design of a Family of Products Through the Use of the Robust Concepts Exploration Method," *6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, WA, AIAA, Vol. 2, September 4–6, pp. 1535–1545.
[10] Simpson, T. W., Chen, W., Allen, J. K., and Mistree, F., 1999, "Use of the Robust Concept Exploration Method to Facilitate the Design of a Family of Products," *Simultaneous Engineering: Methodologies and Applications* (Roy, U., Usher, J. M., et al., eds.), Gordon and Brach Science Publishers, Amsterdam, The Netherlands, pp. 247–278.
[11] Messac, A., Martinez, M. P., and Simpson, T. W., 2002, "Effective Product Family Design Using Physical Programming and the Product Platform Concept Exploration Method," *Engineering Optimization*, Vol. 3, (in press).
[12] Messac, A., 1996, "Physical Programming: Effective Optimization for Computational Design," AIAA J., **34**(1), pp. 149–158.
[13] Messac, A., 1994, "Physpro: A Matlab-based Physical Programming Software," Optimal Systems, 18 Winchester Drive, Lexington, MA.
[14] Messac, A., Sundararaj, G. J., Tapetta, R. V., and Renaud, J. E., 1999, "The Ability of Objective Functions to Generate Non-Convex Pareto Frontiers," Paper # AIAA-99-1211, *40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, St. Louis, MO, April 12–15.
[15] Messac, A., "From the Dubious Construction of Objective Functions to the Application of Physical Programming," AIAA J., (Exp. pub. Date, Jan. 2000).
[16] Messac, A., Gupta, S., and Akbulut, B., 1996, "Linear Physical Programming: Effective Optimization for Complex Linear Systems," *Transactions on Operational Research*, October, **8**, pp. 39–59.
[17] Messac, A., and Hattis, P., 1996, "Physical Programming Design Optimization for High Speed Civil Transport (HSCT)," AIAA J. **33**(2), pp. 446–449.
[18] Messac, A., and Wilson, B., 1998, "Physical Programming for Computational Control," AIAA J., **36**, pp. 219–226.
[19] Messac, A., and Chen, X., 2000, "Visualizing the Optimization Progress in Real-Time Using Physical Programming," *Engineering Optimization Journal*, Vol. 32, No. 5.
[20] Messac, A., Melachrinoudis, E., and Sukam, Cyriaque, P., 1999, "Physical Programming: A Mathematical Perspective," *38th Aerospace Sciences Meeting & Exhibit, Reno, NV*, Paper No. AIAA 2000-0686.
[21] Messac, A., and Sundararaj, G., 1999, "A Robust Design Approach Using Physical Programming," *38th Aerospace Sciences Meeting & Exhibit, Reno, NV*, Paper No. AIAA 2000-0562.
[22] Chapman, S. J., 1991, *Electric Machinery Fundamentals*, McGraw-Hill, New York.
[23] Veinott, C. G., and Martin, J. E., 1986, *Fractional and Subfractional Horsepower Electric Motors*, McGraw-Hill, New York.
[24] Lehnerd, A. P., 1987, "Revitalizing the Manufacture and Design of Mature Global Products," *Technology and Global Industry: Companies and Nations in the World Economy* (Guile, B. R. and Brooks, H., eds.), National Academy Press, Washington, D.C., pp. 49–64.
[25] G. S. Electric, 1997, "Why Universal Motors Turn on the Appliance Industry," http://www.gselectric.com/products/univers14.asp
[26] Simpson, T. W., Maier, J. R. A., and Mistree, F., 1999, "A Product Platform Concept Exploration Model for Product Family Design," *Design Theory and Methodology—DTM'99*, Las Vegas, Nevada, ASME, September 12–15, Paper No. DETC99/DTM-8761.