2006

SPPBRD: A decision framework for scalable product platform based robust design.

Caiyun Wang
University of Windsor

SPPBRD: A DECISION FRAMEWORK FOR SCALABLE PRODUCT PLATFORM BASED ROBUST DESIGN

by

Caiyun Wang

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through Industrial and Manufacturing Systems Engineering
in Partial Fulfilment of the Requirements for the
Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2006

Your file Votre référence
ISBN: 978-0-494-17058-8
Our file Notre référence
ISBN: 978-0-494-17058-8

ABSTRACT

Today's competitive and highly volatile market is redefining the way companies do business. A main competitive advantage for many companies is the ability to bring the products to market faster. An effective method to get the advantage is to develop product platforms. This thesis develops a methodology to assist companies in creating product platforms quickly and efficiently. The thesis focuses on building a model for scalable product platforms and developing a framework for the Scalable Product Platform Based Robust Design since there are many researches on module-based product platform and no systematic framework for scalable product platform.

In the methodology, the two-stage approach, multiple-objectives, compromise decision support problem (compromise DSP), and robust design are integrated to build the decision model. The model consists of eight steps that describe how to formulate the problem and how it can be solved.

The methodology is divided into two stages. The first stage is to build an optimization model and solve this model to get the common product platform in which the design variables can be kept as constants. The second stage is to instantiate the individual products and then to create the product platform. The key role of the first stage is to develop the compromise DSP, a flexible decision support construct that facilitates the search for satisfying compromises among multiple, conflicting goals. The compromise DSP also accommodates multiple constraints and bounds on the system variables and can be implemented with reasonable effort. The compromise DSP model can be transferred into a mathematical optimization model.

The essential of the methodology is to infuse the robust concept into the product platform design. Two tasks of robust design, achieving performance targets and minimizing performance variation are used in the compromise DSP model perfectly, in which achieving performance targets is described as "mean on target", and minimizing performance variance is used as a commonality goal.

Testing and verification of the method occurs through the design of a tube-fin evaporator platform that is scaled around the design variables.

ACKNOWLEDGMENTS

"I believe that friends are quiet angels who sit down on our shoulders and lift our wings when we forget how to fly."

-Unknown

There are many such angels who have been in my life and my world during these last two years. I want to thank every angel who has helped me and made a positive impact on my life during these days. First and foremost I offer my deepest thanks to my advisor, Dr. Michael H. Wang, I would like to thank him for his extraordinary support, supervision, suggestion, enthusiasm and inspiration over the past two years at University of Windsor, without whom I would not be considering the idea of continuing further in academics. I owe him lots of gratitude for having me shown this way of research in which I enjoyed very much.

I am also thankful to my committee members, Dr. Amir Fartaj, Dr. Guoqing Zhang, and Dr. Walid Abdul-Kader for their valuable suggestions and critics.

I would like to give my sincere thanks to Ms. Jacquie Mummery and Mr. Ram Barakat, not only for their daily assistance and support, but also for their friendship.

I wish to thank all people around me that have made this a memorable journey. I give special thanks to Sumita Pangpaonisila and Helen Zhang who have been great friends and guide. Thanks, my friends, for all your encouragement and support.

I offer my deepest gratitude to my parents to whom I owe everything. I dedicate this thesis to them. I love you very much. I give heartfelt thanks to my sisters, brother and my husband, who have lifted my spirits whenever I was down and encouraged me to do better. I would not be writing these lines without the unconditional love and support of my family.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

The objective in this thesis is to develop the decision model for scalable product platform based robust design to facilitate the design of a common scalable product platform that aims to satisfy a range of performance (or dimensions and other parameters) requirements using the smallest variation of the product designs. The chapter 1 gives the foundations for product family and product platform design. The heart of chapter 1 lies in section 1.3 wherein the thesis research objectives are described. Section 1.1 contains the background and motivations of the product family and product platform design including some examples of successful product families, some definitions and opportunities for advancing this thesis. In section 1.2 the foundations for the Decision-Based Design and robust design are presented. Finally, the organization of the thesis is contained in Section 1.4.

1.1 Background and Motivations

1.1.1 Customer-driven market's requirements

Today's competitive and highly volatile market is redefining the way companies do business. "Customers can no longer be lumped together in a huge homogeneous market, but are individuals whose individual wants and needs can be ascertained and fulfilled" (Pine, 1993). Companies are being called upon to deliver better products faster and at less cost for customers who are more demanding in a market that is characterized by words such as mass customization and rapid innovation. Even government agencies like NASA are re-examining the way they operate and do business and adopt slogans such as "better, faster, cheaper." The basic principle in design is to get a quality product

to market quickly and then remain competitive in the marketplace through continuous development of a product line.

1.1.2 Traditional design methods face new challenge

Usually, a long-term success of an enterprise depends on a stream of new products – some replacing older ones, others pioneering new markets. Regardless of the importance creating streams of new products, traditional methods for designing new products and managing this vital business function usually fail to deliver in the long run. Many companies focus on new product identification without corresponding attention to maximize the existing product systems, Benchmarking shows that fewer than 10% of companies have fully embraced all the key components of a robust product family and platform lifecycle management approach. The single-product focus is a failure to embrace commonality, compatibility, standardization or modularization among different products and product lines.

Fortunately, today's most companies know that long-term success does not hinge on any single product, but on a continuous stream of value-rich products that target growth markets step by step. They have found that cost efficiencies, technological leverage, and market power can be achieved when they redirect their thinking and resources from single products to families of products built upon robust product platforms.

At the same time, companies are being faced with the challenge of providing as much variety as possible for the market with as little variety as possible between products. How to solve this conflict to satisfy the manufacturers and customers? One of solutions is to design and develop a family of products with as much commonality between products

as possible with minimal compromise in quality and performance. Gratifyingly, many companies are adopting the concept of product families to improve customization for today's stern competitive global marketplace and at the same time to cater to customer's requirements.

1.1.3 Fuzzy front end (FFE) for product family and product platform

The FFE is defined by those activities that come before the more formal and well-structured New Product Development (NPD) process (Koen, et.al., 2002). Even though there is a continuum between the FFE and the new product development, the activities in the FFE are often chaotic, unpredictable and unstructured (Peter A. Koen , 2002)

The figure 1.1 shows the schematic of the "typical" five-stage five-gate model.

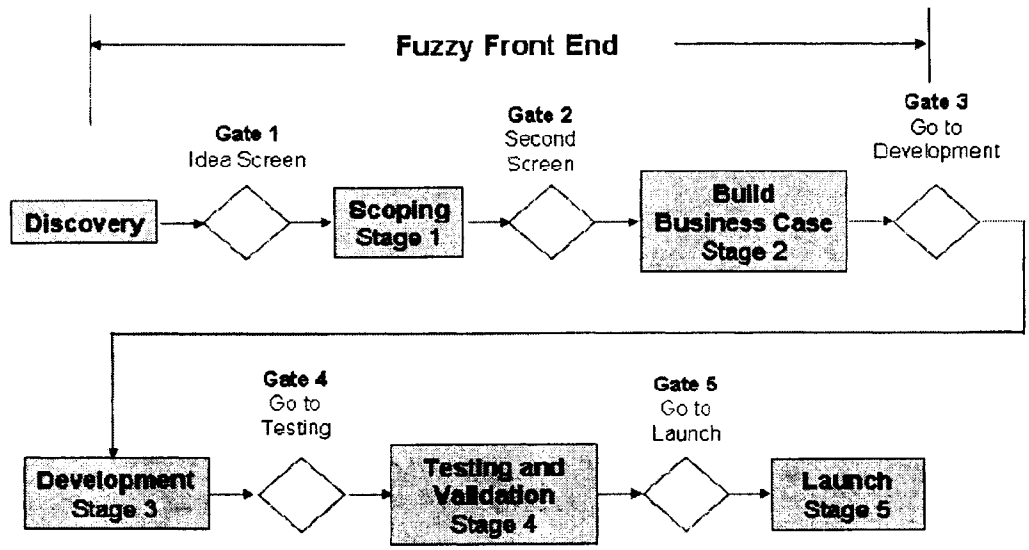

Figure 1.1 Typical five stage, five gate model of Stage Gate™ (Winning at New Products, 2001)

Typically the platform plan, with its first product is evaluated at Gate 3, with subsequent incremental extensions following the traditional Stage Gate™ process. The overall process typically is an intensive effort that involves 3 -5 people for often as much as 6 months. Though the project can often be shortened to 2 – 3 months if many of the members of the team are committed on a full time basis (McGrath, 2001)

Developing a platform and accompanying product strategy based on the strategic vision typically is done in the following 4 chronological steps. This effort should not be undertaken until there is consensus between the team and senior management on the strategic vision. The four steps are:

1. Segmenting and understanding the market.

Before specific concepts can be developed the platform team needs to clearly understand how the market is segmented, the unmet customer needs in and strength of the competitors within each segment.

2. Developing initial product concepts.

Product concepts that satisfy the needs and build on the core competencies, capabilities or channels of the company. A concept is not a product, but a well-defined form including both a written and visual description, which includes its primary features and customer benefits combined with some understanding of the technology needed (Koen, et. al. 2002). A product concept for the Black and Decker example could consist of rough sketches of a common motor and how it could integrate and be part of drills, sanders and circular saws. Ultimately the product concept needs to build on some unique skills of the company so that a competitive advantage and favorable margins may be achieved.

Multiple product concepts are developed then reevaluated to assess their attractiveness to the market and the company.

3. Developing the product family

Once the initial concepts are determined, a product family with its accompanying product roadmap is developed (Wheelwright and Sasser, 1989). For example HP's Product roadmap of its ink jet printers consisted of its Deskjet (i.e. the initial offering) followed by the Deskjet Plus, the Deskjet writer for Macintosh and then the Deskjet 500, etc.

4. Determining the economic case

Ultimately a business case needs to be developed for the product platform that needs senior management approval. Although the first product released from the product platform may have a negative return on investment since it may have to absorb considerable R&D and operational expenses that are part of the overall platform plan. Traditional "hurdle rate" calculations need to be done on the product family with its stream of products based on a common architecture rather than on the initial offering.

1.1.4 Engineering Examples of Successful Product Families

Product family and product platform have been used in almost every industrial field, in each field they have shown the overwhelming fascination. The following examples from Hewlett-Packard printers, Boeing747, Sony and Black&Decker exemplify successful product families and have been studied as such. Additional examples that might interest the reader include: Xerox copiers (Paula, 1997), Anderson windows (Stevens, 1995), and Kodak single using camera (see, e.g., Clark and Wheelwright, 1993). Volkswagen A-Platform.

HP-Hewlett-Packard printers

The market for home and office computer peripherals---laser printers, ink jet printers, scanners, and various storage devices—has paralleled the burgeoning sales of personal computers. In the early of 1980s, products made by several Asian companies dominated the low-end printer market. Over the course of that decade, however, Hewlett-Packard developed an inkjet product design to establish an expanding beachhead in that market. HP has constantly improved the cost, quality, and speed of its ink jet printers so that they now dominate the low-end market. Its product family renewal has been systematic and vigorous (http://web.cba.neu.edu/~mmeyer/research.html).

The "product family map" for the HP's ink jet printers is shown in the following Figure .The map has a format that we have used many times to portray the evolution and renewal of product families in many industries. Unique platform architecture is defined as the combination of subsystems and interfaces between subsystems that comprise a common product structure for a series of derivative products. The three thickest lines on the map represents the three distinct platform architecture of the inkjet printer product family: the "500" platform, the "600", and the "800" respectively. The lines of medium thickness in Figure 1.2 represent major enhancements to existing platform architecture. These occur when a company replaces one or more existing subsystems in a platform with newer and better technology, all the while maintaining the overall structure or design of the platform. The thinnest of the lines in Figure 1.2 represent specific derivative products based on a product platform. Product family maps quickly reveal the degree to which a firm has both created derivative products from an underlying platform and renewed the platform itself with new designs and component technologies.

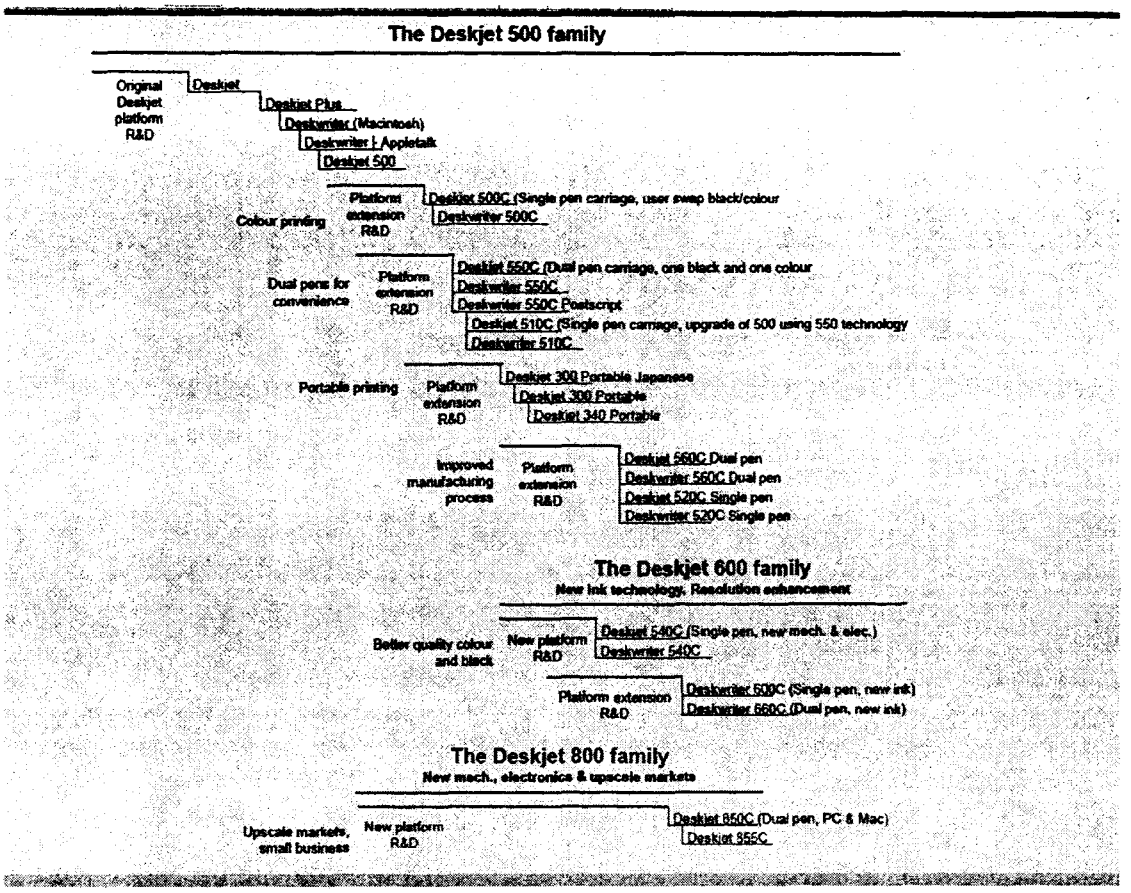

Figure 1.2 The HP Ink Jet Product Family Map (Roger Stake, 2003)

There are compelling management lessons from HP's ink jet story. First, it is a classic case of managing product development from a product family perspective. The company's strategy has been distinctively *tri-modal,* developing derivatives from existing product platforms, enhancing these platforms to address new markets niches or reduce costs, and creating wholly new platforms -- *all at the same time.* Management knew that its competitors (such as Epson) would not acquiesce to its efforts to own the market. Therefore, new generations of inkjet printers would always be required at what is now a breathtaking pace. To bring these innovations to market in a timely manner meant that development work had to be started early. This platform strategy has kept the HP's inkjet

family fresh and competitive, which customers see as a continuous stream of new and increasingly value-rich products. HP has also embraced state of the art manufacturing for its new platform developments. This has made it possible for the company to operate profitably even in a market where complex machinery had to be sold for under $500, and today, well below that price.

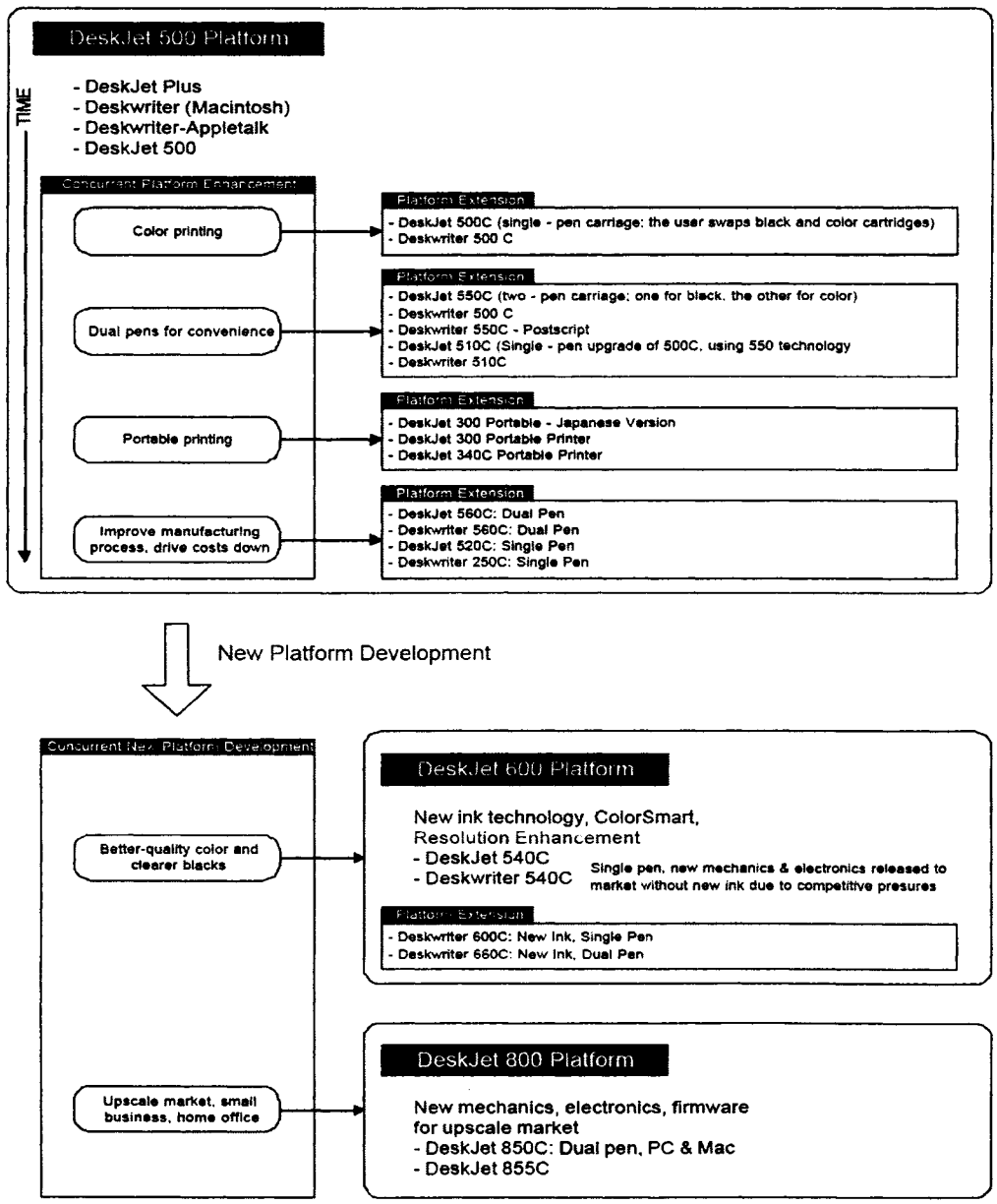

Figure 1.3 The Product Family Map for HP's Ink Jet Printers (Roger Stake, 2003)

N i ppondenso - Automotive Panel Meters

Nippondenso Co. Ltd. supplies automotive components for Toyota, other Japanese carmakers, and carmakers in other countries. They design their panel meters using a combinatorial strategy illustrated in Figure 1.3. A panel meter is composed of six parts (in rare cases, only five), and in order to reduce inventory and production costs, each type 6 of part has been redesigned so that its mating features to its neighbors are identical across the part type. This was done by standardizing the design (denoted by SD in the figure) in an effort to reduce the number of variants of each part. Inventory and manufacturing costs were reduced without sacrificing the product offering. Each zigzag line on the right hand side of Figure 1.2 represents a valid type of meter, and as many as 288 types of meters can be assembled from 17 different components (Simpson, 1999).

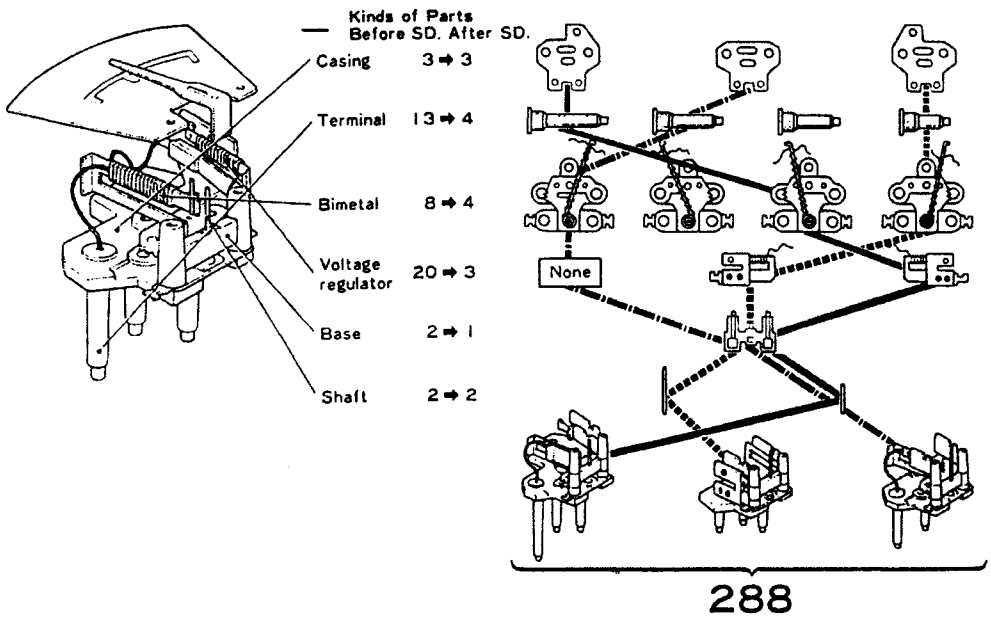

Figure 1.4 Nippondenso Panel Meter Components (from Whitney, 1993)

Volkswagen

As an example, a platform at Volkswagen consists of the floor group, drive system, running gear, along with the unseen part of the cockpit as shown in Fig.1.5. This platform is shared across several models as well as all of its brands (i.e., Volkswagen, Audi, Seat, and Skoda). According to Bremmer (1999), in 1999 Volkswagen owned three of the six automotive platforms that successfully achieved production volumes over one million. The number of million-unit platforms is expected to reach 16 by 2004, with Volkswagen leading the way with its A04 and A4/A5 platforms.

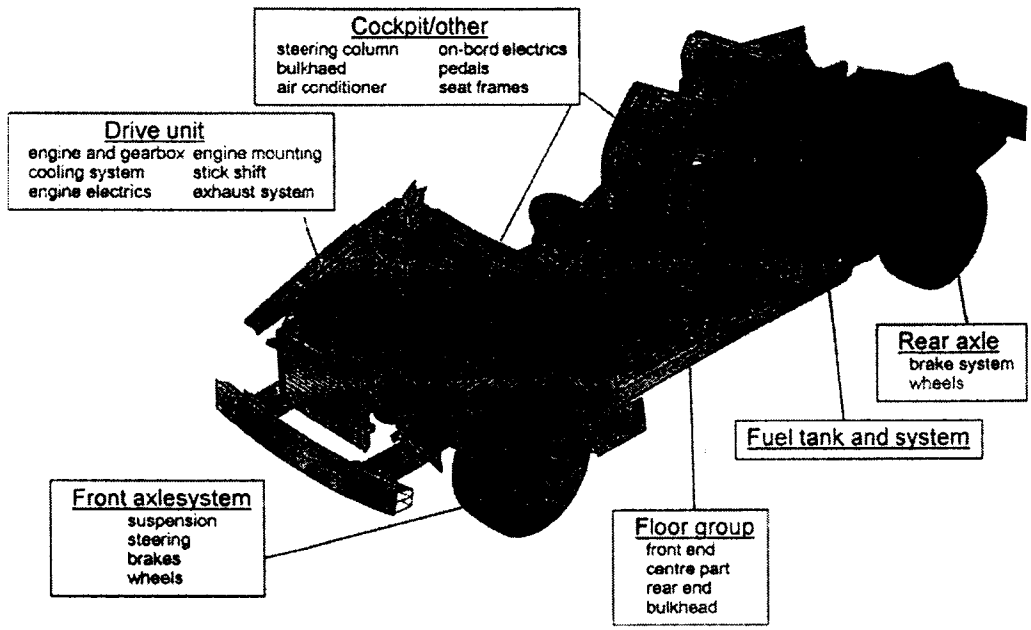

Figure1.5. Volkswagen's Platform (Wilhelm, 1997)

For another example, after working with individual customers to develop 100+ lighting control products, Lutron redesigns its product line around 15-20 standard components that can be configured into the same 100+ models from which customers could initially choose (Pessina & Renner, 1998)

From these examples, we can find the motivation of Product Platform Design:

- Reduce inventory costs
- Reduce design and manufacturing/assembly costs
- Reduce maintenance costs
- Maintain product differentiation

1.1.5 Definitions

In light of these examples, the following definitions for product family, product platform, and derivatives and product variants are offered to provide context for the reminder of the thesis.

Product family is defined as:

- A group of products that share common *form features* and *function(s)*, and target one or multiple market niches. Here, *form features* refer generally to the shape and characterizing features of a product; *function* refers generally to the utilization intent of a product. The Sony Walkman product family is one such example; it contains a variety of models with different features and functions, e.g., graphic equalizer, auto-reverse, and waterproof casing, to target specific market niches (Simpson, 1999).
- A group of related products that share common features, components and subsystems; and satisfy a variety of market niches. A product family comprises a set of variables, features or components that remain constant from product to product (product platform), and others that vary from product to product. The modification of features from product to product within a given family can be effected through scaling (Scale-Based Product Family), or through the addition,

substitution and/or exclusion of modules (Module-Based Product Family) (Achille Messac, 2002).

Product platform can be either narrowly or broadly defined as:

- A set of common components, modules, or parts from which a stream of derivative products can be efficiently developed and launched" (Meyer & Lehnerd, 1997, p. 7)
- A collection of the common elements, especially the underlying core technology, implemented across a range of products (McGrath, 1995, p. 39)
- The collection of assets (i.e., components, processes, knowledge, people and relationships) that are shared by a set of products (Robertson & Ulrich, 1998, p. 20)
- The common set of design variables around which a family of products can be developed. In general terms, a product platform is the common technological base from which a product family is derived through modification and instantiation of the product platform to target specific market niches (Simpson, 1999).
- The set of parameters (common parameters), features, and/or components that remain constant from product to product within a given product family (Achille Messac, 2001)

A product platform describes an architecture used to develop a family of products. The architecture allows for sharing of components, subassemblies, assembly sequences, etc. between product variants. Without a product platform, products are designed for individual performance (Ryan Fellin, 2001).

There are other terms defined as following:

Common Parameters: design parameters that remain constant from product to product within a given product family, which constitute the product platform.

Module-Based Product Family: product family in which features change from product to product through the addition, substitution and/or exclusion of modules.

Derivative or **product variant:** a specific instantiation of a product platform within a product family that possesses unique form features and function(s) from other members in the product family. Paper copiers are good examples of products derived from a common product platform; in addition to the Canon example discussed previously, Xerox's 1090 copier is a derivative of its 1075 model while both copiers are part of Xerox's 10 series of copiers (Jacobson and Hillkirk, 1986). Furthermore, the Boeing 747-200, 747-300, and 747-400 are derivatives of the Boeing 747 (Rothwell and Gardiner, 1990).

Single product or **individual product:** a unique product that has no pre-defined relationships to other products; any resemblance to other products is strictly through coincidence or producer's preference (Erens, 1997). A single product contrasts a derivative product that has similarities to other products in the product family having been derived from the same product platform.

The key to a successful product family is the product platform from which it is derived either by adding, removing, or substituting one or more modules to the platform or by scaling the platform in one or more dimensions to target specific market niches (Simpson, 2003).

1.1.6 Opportunities in product platform

To understand some of the research opportunities in product family and product platform design, a closer look at the previous examples is necessary. The examples from Hewlett Packard printer, Nippondenso Automotive Panel Meters, Volkswagen exemplify a bottom-up approach to product family design. Each company redesigned or consolidated a group of distinct products to create a more efficient and effective product family.

The main objective for this approach is to simplify the product offering and reduce part variety by standardizing components so as to reduce manufacturing costs and inventory costs and reduce manufacturing variability (i.e., the variety of parts that are produced in a given manufacturing facility) and thereby improve quality and customer satisfaction.

While the cost savings in manufacturing and inventory begin almost immediately from this type of approach, the rewards are typically long-term since the capital investments and redesign costs can be significant (Simpson, 1999).

A company doesn't need to spend millions of dollars in redesign to achieve a good product family. For examples, Rolls Royce, Canon and Sony demonstrate such an approach. They exemplify an a priori or top-down approach to product family design and strategically manage and develop a family of products based on a common platform and its derivatives.

Finally, commonality and standardization across product families allow new designs to be introduced, exploited, and retired with minimal expense related to product development (Lehnerd, 1987).

Good product platforms do not just come off the shelf; they must be carefully planned, designed, and developed. This requires intimate knowledge of customer requirements and a thorough understanding of the market. However, as discussed in the literature review in Section 2.1.1, many of the tools and methods which have been developed to facilitate the management and development of effective product platforms and product families are at too high of a level of abstraction to be useful to engineering designers particularly for modeling and design synthesis. Meanwhile, engineering design methods and tools for synthesizing product families and product platforms are limited or slowly evolving. Table 1.1 summarizes some approach and available support for the examples in this section and some organizations that use product platforms.

Examples	Top-Down or Bottom-up	Productivity Family Composition	Availability of Design Support
HP-Hewlett Packard	Top-Down	Product platform which is both scaled and modular for upgrading	Modular and scalable design
Lutron: Lighting Control Systems	Bottom-Up	Combinatoric strategy based on modular design and part standardization	Modular design and clustering approaches
Nippondenso	Bottom-Up	Similar to Lutron	Clustering approaches and modular design
Volkswagen	Bottom -Up	Similar to Lutron	Modular design
Sony: Walkman	Top-Down	Product platform with Predominantly modular design innovations	Modular design
Black & Decker: Universal Motor	Bottom-Up	Product platform scaled around stack length	Modular design to standardize interfaces.
Rolls Royce: RTM322 Engine	Top-Down	Product platform which is both scaled and modular for upgrading	Modular design for some components.

Table 1.1. Product Family Examples: Approach and Available Support

The main task of a product family designer is to decide the right components/design variables to share among products to maintain economies of scale with minimum sacrifice in the performance of each product in the family (Jaeil Park, 2004). Then there are two directions to design product family: design the right components or design variables; the former uses the method modular design, and the latter correspondingly close to variable design.

The prominent approach to platform-based product development top-down or bottom-up is through the development of a *Module-Based Product Family* wherein product family members are instantiated by adding, substituting, and/or removing one or more functional modules from the platform. An alternative approach is through the development of a *Scale- Based Product Family* wherein one or more scaling variables are used to "stretch" or "shrink" the platform in one or more dimensions to satisfy a variety of market niches (Simpson, 2003).

The majority of the examples from this table require modular design to facilitate upgrading and derating product variants through the addition and removal of modules; a survey of these many of approaches is offered in chapter 2 In addition, clustering approaches have been developed to reduce variability within a product family and facilitate redesigning product families to improve component commonality.

Meanwhile, *little attention has been paid to platform scaling issues for product family design.* The notion of a "scalable" or "stretchable" product platform is introduced by Rothwell and Gardiner (1990) and may be loosely defined as follows:

Scalable refers to the capability of a product platform to be "scaled," "stretched," or "leveraged" to satisfy specific market niches. For example, the Boeing 747 is a scalable product platform. It has been "scaled up" and "scaled down" to create the

Boeing 747-200, 747-300, and 747-400 to satisfy different market niches based on number of passengers, flight range, etc. (Rothwell and Gardiner, 1990). The Rolls-Royce RTM322 aircraft engine and the Black & Decker universal motor examples discussed in Section 1.1.1 heavily exploit platform scaling.

Scaling Variables: design variables that vary from product to product within a given product family. Scaling variables can be used to ''stretch'' or ''shrink'' members of the product family to instantiate their individual performance.

Scalable Product Family: product family in which features change from product to product through different values of the scaling variables.

There are several reasons to investigate scalability in product platform design:

While modular design has received considerable attention in engineering design research, the design of parametrically scalable product platforms for a product family has received little to none.

In many product families, scalability can be exploited from both a technical standpoint and a manufacturing standpoint to increase the potential benefits of having a common product platform. The Rolls Royce RTM322 engine and the Black & Decker universal motor are excellent examples of this.

Finally, and perhaps most importantly, the concept of scalability and scalable product platforms provides an excellent inroads into product family and product platform design through the synthesis of current research efforts in Decision-Based Design and the

Robust Concept Exploration Method (described in Sections 1.2.1 and 1.2.2, respectively), robust design (described in Section 2.3) and tools from marketing/management science (described in Section 2.1.2).

The main task of a product family designer is to decide the right components/design variables to share among products to maintain economies of scale with minimum sacrifice in the performance of each product in the family.

The emphasis in this paper is on scale-based product family and formulation of the resulting product family optimization problem used to design the product platform and corresponding scale-based product platform. The foundation for developing this approach is presented in the next section. The specific research focus for the thesis is outlined in Section 1.3.

1.2 Foundations for designing Scalable Products Platforms for a Product Family

The technology base for the dissertation is described in this section. An overview of Decision-Based Design and the compromise Decision Support Problem are given in Section 1.2.1. This is followed by an overview of Robust Concept Exploration Method (from which the product platform concept exploration method is derived).

1.2.1 Decision-Based Design, Decision Support Problem Technique, and the Compromise DSP

Decision-Based Design (DBD) is rooted in the notion that the principal role of a designer in the design of an artifact is to make decisions (see, e.g., Muster and Mistree, 1988). This role is useful in providing a starting point for developing design methods

based on paradigms that spring from the perspective of decisions made by designers (who may use computers) as opposed to design that is predicated on the use of computers, optimization methods (computer-aided design optimization), or methods that evolve from specific analysis tools such as finite element analysis.

The implementation of Decision-Based Design is the Decision Support Problem (DSP) Technique, a technique that supports human judgment in designing systems that can be manufactured and maintained. In the DSP Technique, designing is defined as the process of converting information that characterizes the needs and requirements for a product into knowledge about a product (Mistree, et al., 1990). This definition is extended easily to product family design: the process of converting information that characterizes the needs and requirements for a product family into knowledge about a product family, or as is the case of this work, a common scalable product platform. A complete description of the DSP Technique can be found in, e.g., (e.g., Mistree, et al., 1990). Mistree gave a brief history of the development of algorithm and the development of the DSPs. The compromise DSP is derived from goal programming.

In goal programming a distinction is made between an objective and a goal:

Objective: In mathematical programming, an objective is a function that we seek to optimize, via changes in the problem variables. The most common forms of objectives are those in which we seek to maximize or minimize. For example, Minimize $Z = A(X)$

Goal: It is an objective with a "right hand side". This right hand side (G) is the target value or aspiration level associated with the goal. For example,

$A(X) = G$

In a **goal** we can distinguish the aspiration level, G_i, of the decision maker and the actual attainment, Ai (X), of the goal. Three conditions need to be considered:

1. $A_i(X) \leq G_i$ We wish to achieve a value of Ai (X) that is equal to or less than G_i.

2. $A_i(X) \geq G_i$ We wish to achieve a value of Ai (X) that is equal to or greater than G_i.

3. $A_i(X) = G_i$ We would like the value of Ai (X) to equal G_i.

We will now introduce the concept of a **deviation variable**. Consider the third condition; namely, we would like the value of Ai (X) to equal G_i. The deviation variable is defined as:

$$d = G_i = A_i(X)$$

The deviation variable d can be negative or positive (d_i^-, d_i^+), representing under-achievement or over-achievement of each goal with respect to target value, G_i; but in engineering applications, we prefer the term deviation function instead of achievement function. We consider this term to be more appropriate, since the function provides us with a measure of the deviation from the goals.

In effect, a deviation variable represents the distance (deviation) between the aspiration level and the actual attainment of the goal. Considerable simplification of the solution algorithm is effected if one can assert that all the variables in the problem being solved are positive. Hence, the deviation variable d is replaced by two variables:

$$d = d_i^- - d_i^+ \qquad \text{Where } d_i^- \cdot d_i^+ = 0; d_i^-, d_i^+ \geq 0$$

The preceding ensures that the deviation variables never take on negative values. The product constraint ensures that one of the deviation variables will always be zero.

The system goal becomes:

$$A_i(X) + d_i^- - d_i^+ = G_i; i = 1,2,3....m$$

Subject to $d_i^-, d_i^+ \geq 0$ and $d_i^- \cdot d_i^+ = 0$

The basic model can be described in Figure 3.7.

In effect the traditional formulation is a subset of the compromise DSP – an indication of the generality of the compromise formulation. The compromise DSP is stated in words as follows:

The solution of the compromise DSP is a feasible point that achieves the system goals to the best extent that is possible. This notion of satisfying solutions is in philosophical harmony with the notion of developing a broad and robust set of top-level design specifications. Developing ranged sets of top-level design specifications is generalized into the notion of developing a *product platform portfolio*. By finding a "portfolio" of solutions rather than a single point solution, greater design flexibility can be maintained during the design process.

The compromise DSP is a flexible decision support construct that facilitates the search for satisfying compromises among multiple, conflicting goals. It also accommodates multiple constraints and bounds on the system variables and implemental with reasonable effort. It is domain independent. Also, the compromise DSP is applicable along a design timeline, including during the early stages of design or under other conditions when decisions must be made quickly and/or with limited information (Williams, 2003).

Finally, the compromise DSP also provides the cornerstone of the Robust Concept Exploration Method that will be reviewed in the next section.

1.2.2 Product platform based robust design

In a competitive market, a product quality affects manufacturer's status. In order to achieve a high product quality, a successful product design is a must. During the product development process, a great deal of uncertainties exists. The uncertainties, such as changes in customer needs, changes in technological developments, and existence of competitors, may affect the process of designing a product. Hence, a good product design must provide an additional flexibility to allow quick changes in a product design (Apichat Sopadang, 2000).

To increase the flexibility of the product design, the conventional robust design may be effectively integrated with the concept of product family. By combining the basic concept of Robust Design with the notions of Product Family and Product Platform Design, high quality products with a low cost can be produced while maximizing market leverage from the common technology and common product platform. At the same time, with the principle of robust design, the quality of a product can be improved by minimizing the effect of the causes of variation without eliminating the causes.

Taguchi first proposed the concept of robust design that stressed on improving the quality of a product or process by not only striving to achieve performance targets but also by minimizing performance variation. Taguchi's methods have been widely used in industry (Byrne and Taguchi, 1987; Phadke, 1989) for parameter and tolerance design (Rakesh S.Kilkarn, 2005)

Robust design is an engineering methodology for optimizing the product and process conditions which are minimally sensitive to the various causes of variation, and which produce high-quality products with low development time and manufacturing cost. Although this design method has clearly been proven important for many industries, there is a significant room for improvement. The major difficulty associated with implementing the current robust design principles for real-world industrial problems is a lack of consideration of a family of products- a group of related products when designing products and processes. The Product Family concept that allows the flexibility to the design system should be integrated with the concept of Robust Design, which is *Product Family Based Robust Design (PFBRD).*

The question then becomes how to integrate robust design with the product family concept. A PFBRD is a comprehensive engineering methodology, consisting of procedures for carrying out the following 6 steps:

Step 1 – Identification of quality attributes and corresponding factors: This step involves the procedure to classify different design parameters (control factors, and noise factors) and their responses (quality attributes).

Step 2 – Assessing weights to quality attributes: Based on the fact that product design is a multiple quality attributes problem, the assessing weight to each quality attribute is an inevitable task. The integrating of multiple attribute decision-making, classical assessing weight method such as eigenvector and entropy method, fuzzy sets, and Monte Carlo simulation can be used for this purpose.

Step 3 – Identification of scaled factor(s): A variety of products can be created from common product family platform using stretch design and scaled factor. Thus, after

identify design variable, quality attributes and it corresponding weight in Step 1 and 2, scale factor should be identified and integrated to conventional robust design concept. Robust design, conceptual robustness, product family design, and multiple attribute decision-makings can be used to identify the scaled factor. Subsequently, this scaled will be used in Step 4 to construct and experimental design for PFBRD.

Step 4 – Design of experiments for PFBRD: To make the product platform flexible and robust, a modification in conventional experimental design is needed. Scaled factors should be integrated to the experimental design table. As a result, an experimental design that concern of multiple responses based on the effect of control factors, noise factors, and scaled factor. Next, response surface methodology can be used to identify key design drivers and the significance of different design factors, and to analyze the result.

Step 5 – Optimization of multiple quality attributes: After constructed multiple responses using RSM, multiple objective decision making technique can be used to find the best setting of design parameters for common product platform that is less sensitive to noise factors and scaled factors. Consequently, a high quality and robust product platform that is consist to variations in production process and operating environment. In addition, the common product platform also provides an enhance flexibility for making change regarding to customers, market niches, technologies, and government regulation.

Step 6 – Creation of product platform: The robust common product platform can be constructed based on the result of optimization result

1.3 Research Objectives

The principle goal in this thesis is develop a decision model to facilitate the design of a scalable product platform around which a family of products can be developed. As discussed in previous section, Decision-Based Design and Robust Concept Exploration Method provide the foundation on which this work is built. Given this foundation and goal, the research objectives are as follows:

1. Develop a framework to complete the decision model for common scalable product platform and its individual products step by step
2. Robust design principles can be used to facilitate the design of a common scalable product platform by minimizing the sensitivity of a product platform to variations in scale factors
3. Individual targets for product variants can be aggregated into an appropriate mean and variance and used in conjunction with robust design principles to affect a common product platform for a product family
4. Apply the decision model to a practical product platform that can be used in real production

1.4 Organization of the Thesis

The thesis is organized as the following figure1.6 thesis roadmap.

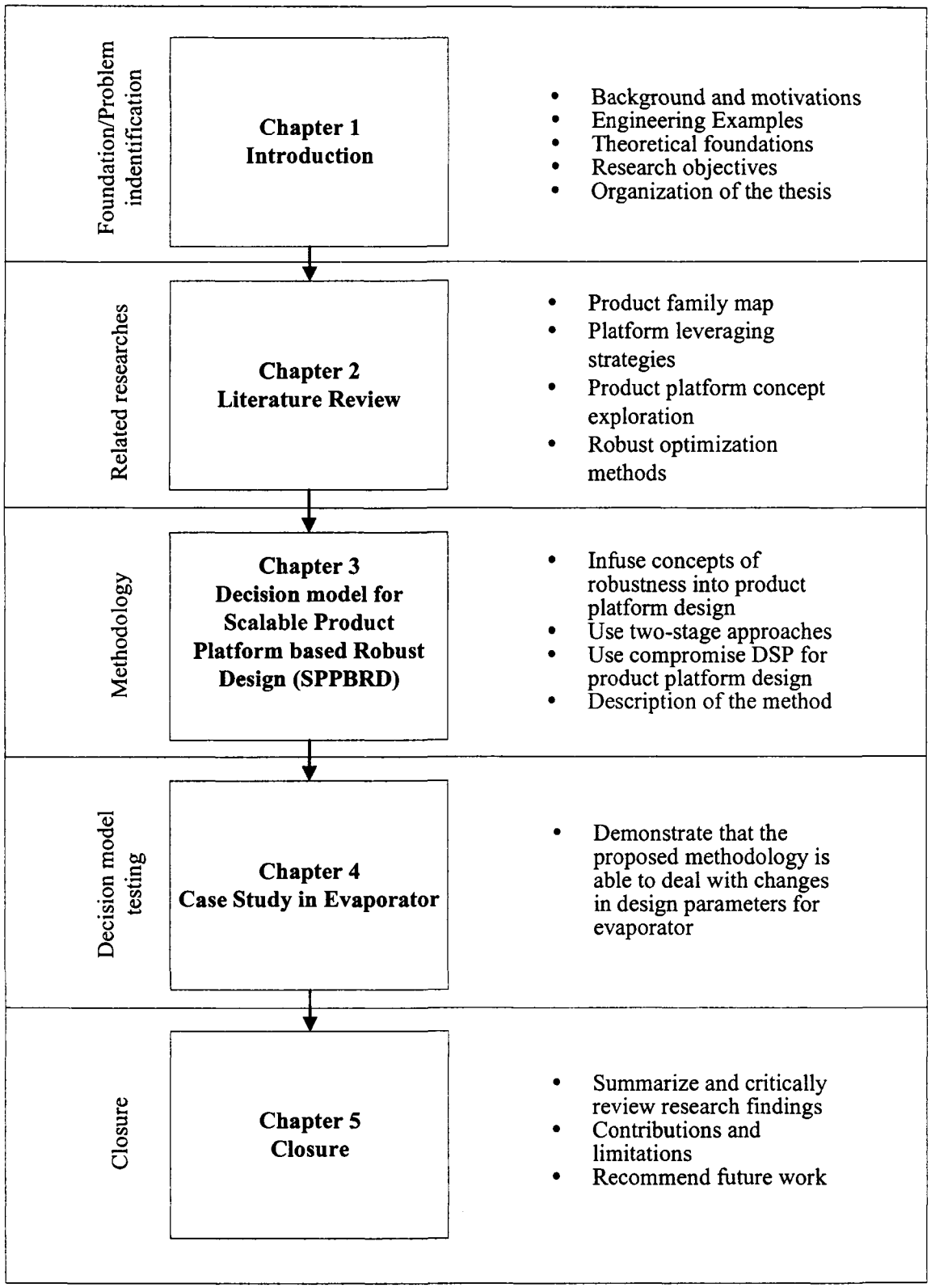

Figure 1.6. Thesis roadmap

CHAPTER 2 LITERATURE REVIEW

This chapter presents a literature survey covering areas of product family; product platform, platform portfolio architectures and product platform based robust design. The chapter concludes by pointing out research gaps and several key issues directly related to the research topic.

2.1 Product Family and Product Platform Design Tools and Methods

According to Pine (1993), "Customers can no longer be lumped together in a huge homogeneous market, but are individuals whose individual wants and needs can be ascertained and fulfilled". Since many companies typically design new products at a time, the focus on individual customers and products often results in "failure to embrace commonality, compatibility, standardization, or modularization among different products or product lines"(Meyer and Lehnerd 1997).

In order to provide as much variety as possible for the market with as little variety as possible between products, many researchers advocate a product platform and product family approach to satisfy effectively a wide range of customer needs.

2.1.1 Product family map

Meyer and Utterback (1993) use the product family map shown in figure 2.1 to trace the evolution of a product family.

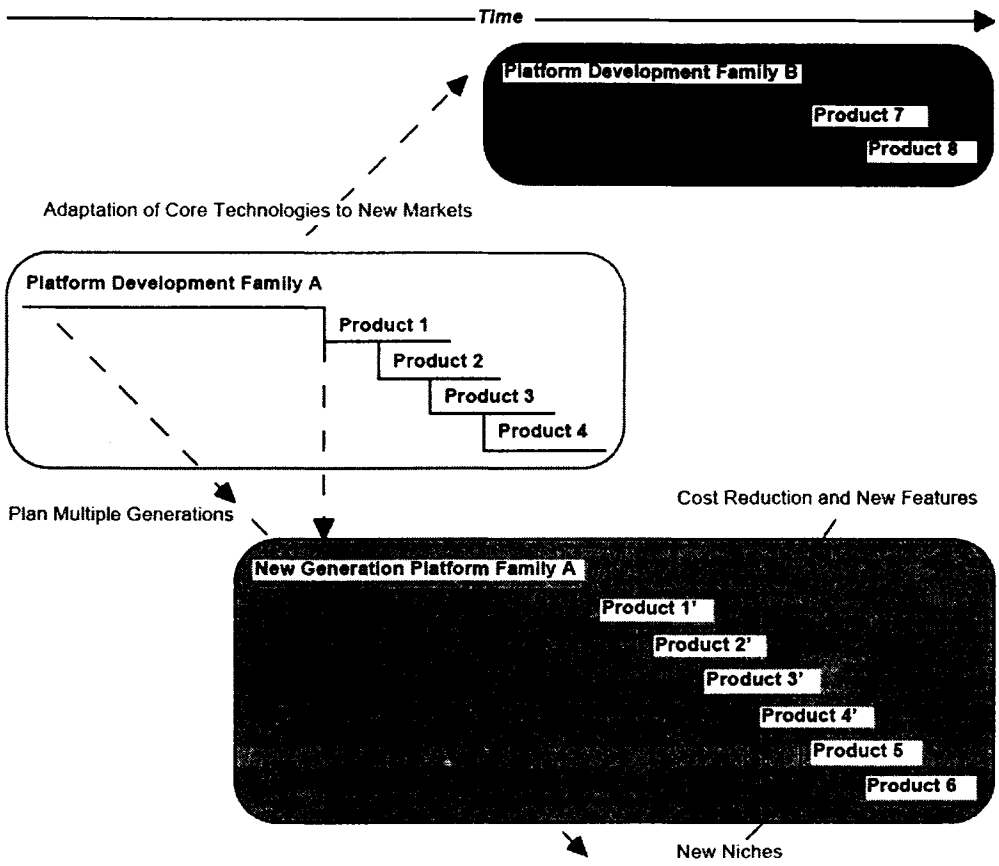

Figure 2.1 Product Family Map (adapted from Meyer and Utterback, 1993)

In their map, each generation of the product family employs a platform as the foundation for targeting specific products at different (or complimentary) markets. Improved designs and new technologies spawn successive generations, and cost reductions and the addition and removal of features can lead to new products. Multiple generations can be planned from existing ones, expanding to different markets or revitalizing old ones. A more formal map, with four levels of hierarchy in the product family (i.e., product family, product platforms, product extensions, and specific products) also is introduced in their work in an effort to assess the dynamics of a firm's core capabilities for product development; several examples can be found in their paper.

In related work, Wheelwright and Sasser (1989) have developed the product development map to trace the evolution of a company's product lines, shown in Figure 2.1. In this map, they also categorize a product line into "core" and "leveraged" product, dividing leveraged product into "enhanced", "customized", "cost reduced" and "hybrid" product.

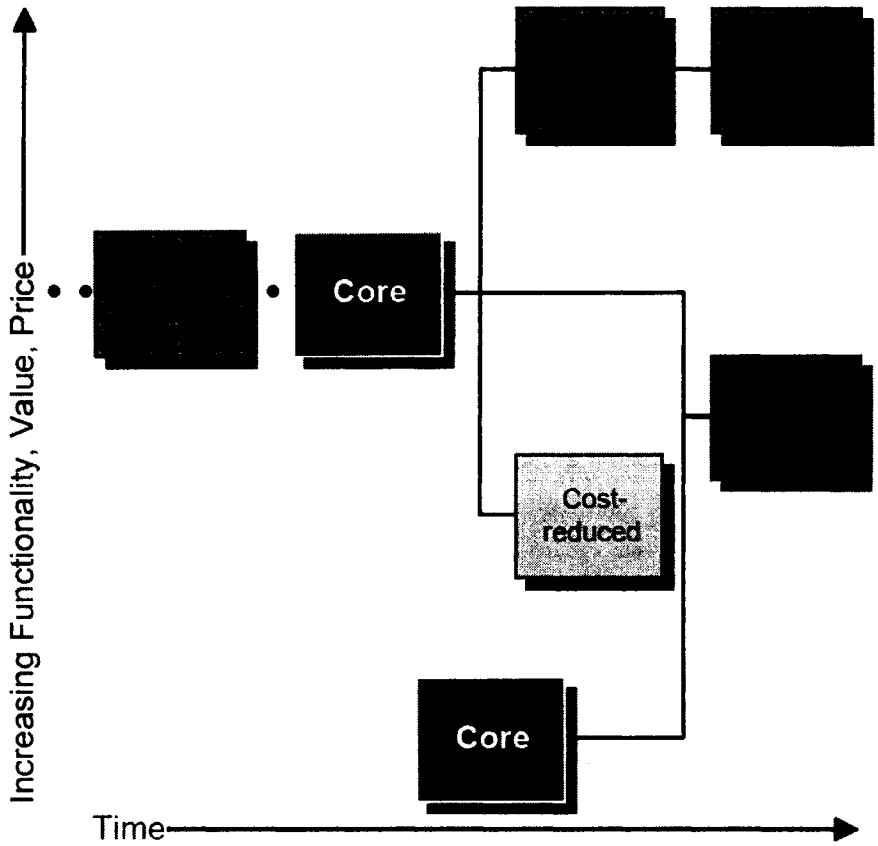

Figure 2.2 Generic Product Development Map (Wheelwright and Sasser, 1989)

As shown in Figure 2.2, the core product, typically derived from an engineering prototype, provides the engineering platform upon which further enhancements are made. Enhanced products are developed from the core by adding distinctive features to target specific market niches; enhanced products are typically the first products leveraged from the core product. Enhanced products can be customized further to provide more choice if

necessary. Cost-reduced products are "scaled" or "stripped" down versions (e.g., less expensive materials and fewer features) of the core, which are targeted at price-sensitive markets.

2.1.2 Platform-leveraging strategies

These product family maps are very useful attention directing tools for product family design and development, but it was not until Meyer [1997] introduced the market segmentation grid that platform-leveraging strategies were clearly articulated. As shown in Fig. 4, market segments are plotted horizontally in the grid while price/performance tiers are plotted vertically; each intersection of a market segment with a price/performance tier constitutes a market niche that is served by one or more of a company's products. Three platform-leveraging strategies can be identified within the grid as shown in Fig. 4: (1) horizontal leveraging, (2) vertical leveraging, and (3) the beachhead approach, which combines both. Meyer and Lehnerd (1997) discuss the advantages and drawbacks of each leveraging approach, and examples of market segmentation grids can be found in (Caffrey, et al., 2002b) for spacecraft and avionics systems and in (Meyer & Lehnerd, 1997) for computers, data storage systems, power tools, and office furniture.

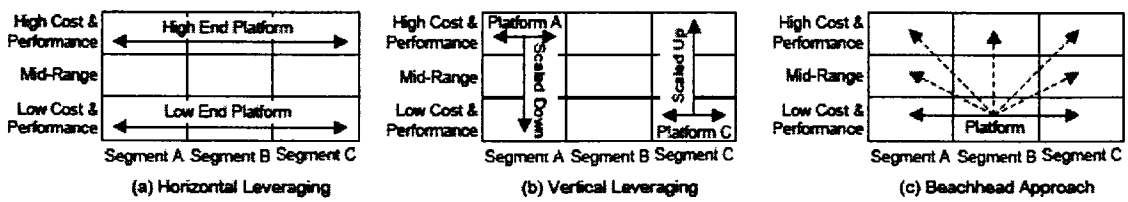

Figure 2.3 Platform Leveraging Strategies (Meyer, 1997)

Timpson in 2003 pointed out that the market segmentation grid is useful for both platform development (i.e., as part of a top-down approach to product family design) as well as product family consolidation (i.e., as part of a bottom-up approach). For instance, Farrell, et al. in 2003 used the market segmentation grid to identify potential platform leveraging strategies for a line of flow control valves using historical sales data. While most horizontal leveraging strategies take advantage of modular platforms, Simpson discusses the relationship between vertical leveraging strategies and scalable platforms in 2001. Finally, Meyer describes adaptations of the market segmentation grid for platform-based development approaches to non-assembled products (Meyer & Dalal, 2002) and the design and renewal of services (Meyer & DeTore, 2001).

2.1.3 Module-Based Product Families

Modularity is an important concept in Product Family; it allows the same component to be used across product variants and production line. By dividing a product into components and interfaces with different desired rates of change, a manufacturer can accommodate necessary change without disrupting the design of entire product. Thus, *Modular Design* is widely practiced and can yield appreciable savings (Apichat Sopadang, 2001).

The prominent approach to product family is the development of Module-Based Product Families wherein product family members are instantiated by adding, substituting and/or removing one or more functional modules from the product platform. Multi-objective optimization approaches for designing families of products are also being developed, with much of the research also focusing on module-based product families. For instance, Nelson formulate the product platform design problem using multi-criteria

optimization to resolve the trade-off between commonality and individual product performance within the product family; as an example, they study the Pareto sets of two derivative products (nail guns) to find a suitable product platform. Fujita simultaneously optimize the system structure and configuration of a product family; Fujita extended their previous work by formulating the problem as a 0–1 integer programming problem for modular product architecture development. Gonzalez-Zugasti use a two-stage approach to design a family of spacecraft for three interplanetary missions where each spacecraft consists of 10 subsystems, some of which are shared among all three spacecraft based on the user specified platform. Gonzalez-Zugasti expand their previous work to assess the net present value of a product family using real options to model the risks associated with such factors as uncertainty in technologies and funding.

2.1.4 Scale-Based Product Families

Scaling one or more variables to "stretch" or "shrink" the platform and to create products whose performance varies accordingly to satisfy a variety of market niches develops scale-based product families. This is an alternative approach to product family design. While some consider scale-based product families to be a subset of module-based product families (see, e.g., Fujita & Yoshida, 2001), platform scaling is a common strategy employed in many industries.

This approach is frequently employed in aircraft design, for instance, whereby an aircraft such as the 777-X is ''stretched'' to accommodate an increase in passengers, cargo, or flight range. Automobile manufacturers are also starting to exploit scale-based product families; for example, Honda is developing an automobile platform that can be scaled along its width and length in an effort to realize a ''world car'' (ACHILLE

MESSAC, 2002). Rolls Royce scaled its RTM322 aircraft engine by a factor of 1.8 as shown in Fig.2.4 to realize a family of engines with different SHP (shaft horse power) (Timothy W. Simpson, 2003).

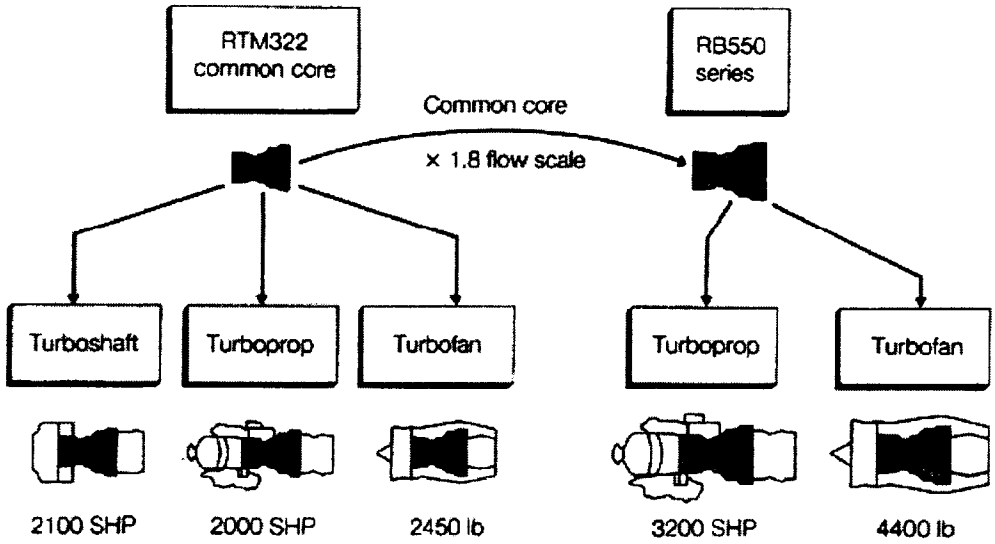

Figure 2.4 A Family of Scale-Based Aircraft Engines (Timothy W. Simpson, 2003)

Previous work in scaled-based family has primarily relied on two-stages approaches wherein the product platform is designed during the first stage, followed by instantiation of the individual products from the product platform during the second stage. Michael P. Martinez in 2001 focused on scale-based product families and presents a new single-stage approach for simultaneously optimizing a product platform and the resulting family of products based on one or more scaling variables – variables that are used to instantiate the product platform by "stretching" or "shrinking" it in one or more dimensions to satisfy a variety of customer requirements. The proposed approach is also unique in that it employs the Physical Programming method, enabling designers to

formulate the product family optimization problem in terms of physically meaningful terms and parameters.

2.1.5 Product Platform Concept Exploration Method

Simpson and Messac proposed the Product Platform Concept Exploration Method (PPCEM) to define the market segment and product specification for a vertically scalable product family in 2001. The steps and associated tools of the PPCEM are shown in Figure 2.3. The input to the PPCEM is the overall design requirements for the set of products, and the output is the set of specifications for the product platform and corresponding family of products. The PPCEM consists of five steps that prescribe how to formulate the problem and describe how it is solved. The actual implementation of each step is likely to vary from problem to problem.

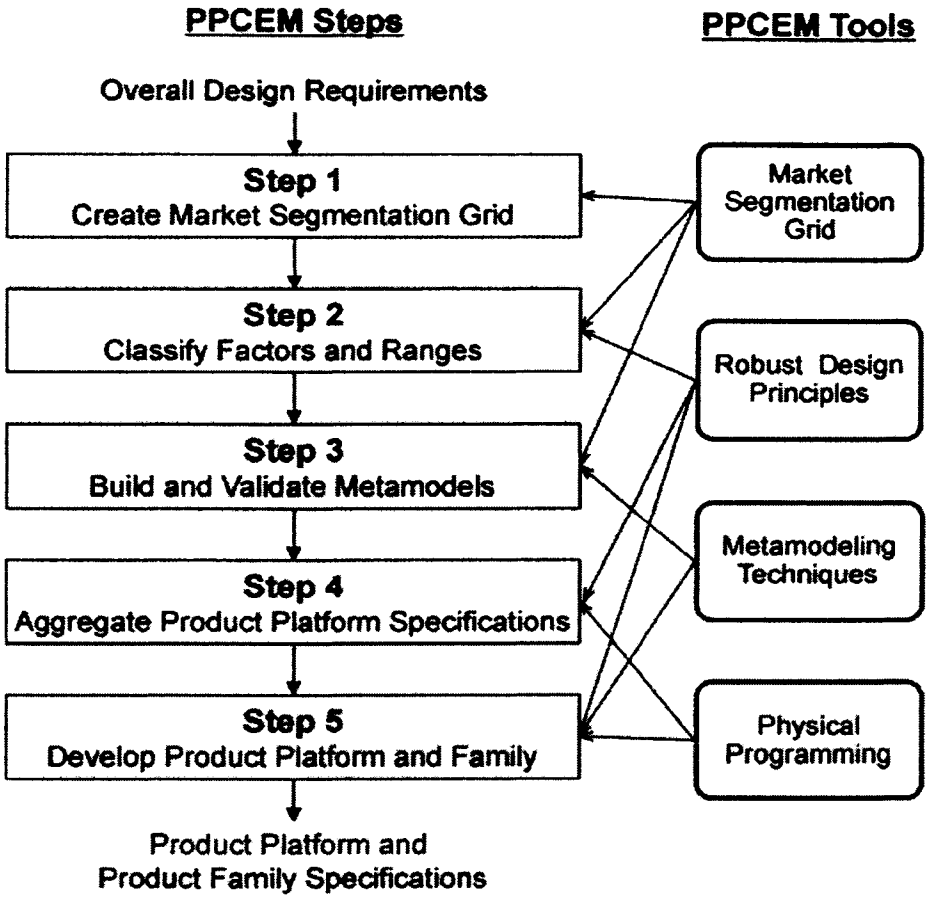

Figure 2.5. Steps and Tools in PPCEM (Achille Messac, 2002)

2.1.6 Top-Down and Bottom-Up approach

There are two basic approaches to product family design (Simpson, et al., 2001a). The first one is Top-Down (proactive platform) approach wherein a company strategically manages and develops a family of product based on a product platform and its derivatives. The second is a bottom-up (reactive redesign) approach wherein a company redesign or consolidates a group of distinct products to standardize components to improve economies of scale. The key to success in either approach is the product

platform around which the product family is derived. Timothy W. Simpson in 2003 listed some successful examples for these two approaches and also gave a summary for definitions for the product platform and product family.

2.2 Optimization-based Approaches

Product family design involves all of the challenges of product design while adding the complexity of balancing the commonality of the products in the family with the individual performance (i.e., distinctiveness) of each product in the family. Multi-objective optimization is experiencing new found use in the field of product family design to help resolve the inherent tradeoff between commonality and distinctiveness (Simpson, 2002).

Multi-objective optimization serves two main purposes during product family design. First, it is *used to help capture the Pareto frontier for a product family.* For instance, Nelson studies the Pareto sets of two derivative products to find a suitable product platform for a family of two nail guns using Multi-objective optimization. Meanwhile, Allada introduces an agent based multi-objective optimization framework to capture the Pareto frontier for module-based product families; he demonstrates his framework using the design of a family of power screwdrivers and electric knives. Second, multi-objective optimization is *used to determine the best design variable settings for the product platform and individual products within the family.* When using multi-objective optimization to determine the best design variable settings for the product platform and individual products within the family, there are two basic approaches that can be summarized as follows (Simpson, T.W and D'Souza, B., 2002):

1. Single-stage approaches – wherein the product platform and resulting family of products are optimized simultaneously;
2. Two-stage approaches – wherein the product platform is designed during the first stage of the optimization, followed by instantiation of the individual products from the product platform during the second stage of the optimization.

Several optimization approaches have been developed within the engineering design community to help determine the best design variable settings for the product platform and individual products within the family.

Kikuo Fujita proposed a simultaneous optimization method for both module combination and module attributes of multiple products. Similarities and differences between different products are explained as shown in Fig. 2.4. That is, different products can share the same modules, and different modules can share some attributes partially.

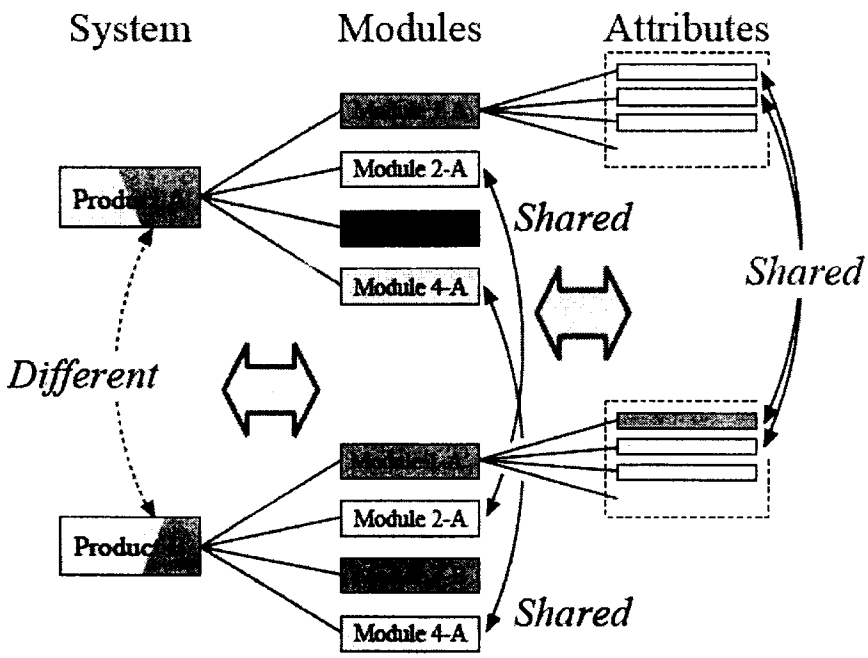

Figure 2.6. System, Modules, and Attributes (Kikuo Fujita, 2001)

In its optimization process, the first is to optimize the combinatorial pattern of module commonality and similarity among different products, the second is to optimize the directions of similarity on scale-based variety, and the third is to optimize the continuous module attributes under the others. Finally it is applied to the simultaneous design problem of multiple airplanes to demonstrate its validity and effectiveness.

2.3 Robust Methods

The robust design objective could be generalized into *two aspects*, namely, "Optimizing the mean of performance" and "minimizing the variation of performance" (Wei Chen, 1998). Current ways of handling multiple aspects using either the Taguchi's signal-to-noise ratio or the weight-sum method are not adequate.

Wei Chen solved bi-objective robust design problems from a utility perspective by the recent development on relating utility function optimization to a Compromise Programming (CP) method. Compared to the existed methods for robust optimization such as Taguchi's signal-to-noise ratio and the weighted-sum method, this approach has capability to generate the efficient solutions, measure utility and is interactive robust design procedure, and offer more flexibility in addressing the multiple aspect of robust design.

Apichat Sopadang, and Byung Rae Cho in 1999 provided a framework for product family based robust design. They proposed 6 steps procedures that are mentioned in section 1.2.2. The methods presented there is a comprehensive system design, which is a hybrid formulation, based on concepts of Robust Design, Product Family Design, Statistical Experiment, Modeling and Simulation Technique, and Optimization. It's capability of creating variety elegant products for customers. Products are less sensitive

to noise and environment, flexible for making change with small cost base on product platform.

Apichat Sopadang, and Byung Rae Cho in 2000 developed a method for assessing weight multiple quality attributes. Two major tools are implemented - fuzzy set theory and Monte Carlo simulation. The fuzzy set theory may be a good means for modeling uncertainty or imprecision arising from environment that human beings are heavily involved in the process of decision analysis. They get three purpose through the investigation: address how to convert qualitative data to quantitative ones by using fuzzy sets, show that simulation can be used to fuzzy sets and demonstrate that customers' and designers' weights of quality attributes can be determined and combined by using a classical assessing weight method such as entropy method and eigenvector. Apichat Sopadang, and Byung Rae Cho also proposed a detailed method for attribute ranking analysis and scaling factors in 1999 and 2000 which are the important parts of the product family based robust design.

2.4 Summary

The following table 2.1 introduces the differences between the literature review above and other studies based on the given aspects. Check mark means that the author did some research on the area. Based on literature review outlined herein, the objectives proposed in Chapter 1 have been formulated. Subsequent Chapters described the methodology used to achieve these objectives.

Research focus / Approach	Module-based family	Scale-based family	Single-objective	Multi-objective	Model manufacturing cost?	Model market demand/sales	Single-stage approaches	Two-stage approaches	Multi-stage approaches	Sequential Linear Programming		Genetic Algorithm	Simulated Annealing		Example product platform (Number of products in platform)
Simpson, et al., 1999		X		X			X								Aviation Aircraft (3)
Simpson, et al., 2001a		X		X				X							Electric motor (10)
Simpson & D'Souza, 2002	X	X		X			X								Aviation Aircraft (3)
Seepersad, et al., 2002		X	X		X	X		X					X		Absorption chillers (12)
Blackenfelt, 2000b	X		X		X		X								Lift tables (4)
Seepersad, et al., 2000		X		X	X	X	X						X		Absorption chillers (12)
Farrell & Simpson, 2003		X	X		X			X							Flow control valves (16)
Nelson, et al., 2001	X			X				X		X					Nail guns (2)
Messac, et al., 2002a		X		X				X				X			Electric motor (10)
Li & Azarm, 2002	X		X	X	X	X	X	X				X			Screwdrivers (3)
Kokkolaras, et al., 2002	X			X				X							Automotive vehicle frame (2)
Hernandez, et al., 2003	X	X	X		X				X						Pressure vessels (16)
Hernandez, et al., 2002		X	X						X						Electric motor (10)
Hernandez, et al., 2001		X		X	X			X					X		Absorption chillers (12)
Fujita & Yoshida, 2001	X	X	X		X	X									Commercial aircraft (4)
Gonzalez-Zugasti & Otto, 2000	X		X			X	X					X			Interplanetary spacecraft (3)
Gonzalez-Zugasti, et al., 2001	X				X		X		X						Interplanetary spacecraft (3)
Fellini, et al., 2000	X			X				X							Flow control valves (16)
Simpson & D'Souza, 2003		X		X			X					X			Aviation Aircraft (3)
Allada & Jiang, 2002	X		X			X			X						Generic modular products (3)
Christopher Williams, 2003		X		X			X			X					Pressure vessels (16)
Rakesh S.Kilkarn, 2005	X	X				X								X	Hand exerciser (5)
Cetin & Saitou, 2003	X			X			X								Weld automotive structures (2)
Fujita, et al., 1998	X		X		X		X						X		Commercial aircraft (2)
Fujita, et al., 1999	X		X		X		X						X		TV receiver circuits (6)
Apichat Sopadang, 2000	X	X												X	
Timothy,Jaeil Parkl,2004	X			X	X							X			

Table 2.1 Difference between others researches and this research

CHAPTER 3 DECISION MODEL FOR SPPBRD

The primary objective in this thesis is to develop a framework for the Scalable Product Platform Based Robust Design (SPPBRD). As seen from the chapter 1 and chapter 2, I wish to integrate two-stage approach, multiple-objectives, compromise decision support problem (compromise DSP), and robust design to create a framework of scalable product platform based robust design. In the current competitive environment, there is a need to embrace commonality, compatibility and standardization among different products and product lines. At the same time, there are changes in customer requirements that make the design parameters change. The purpose of this chapter is to provide theoretical structural validity as shown in Figure 3.1.

The proposed framework of decision model is shown in Figure 3.2. This approach is to integrate robust concept exploration and two-stage approaches into the scalable product platform design to make the entire process robust. The explanation of each step in figure 3.2 is presented in section 3.2.1, stage 1, and section 3.2.2, stage 2.

Before launching into this explanation, it is necessary to first introduce how robust design works for the scalable product platform design.

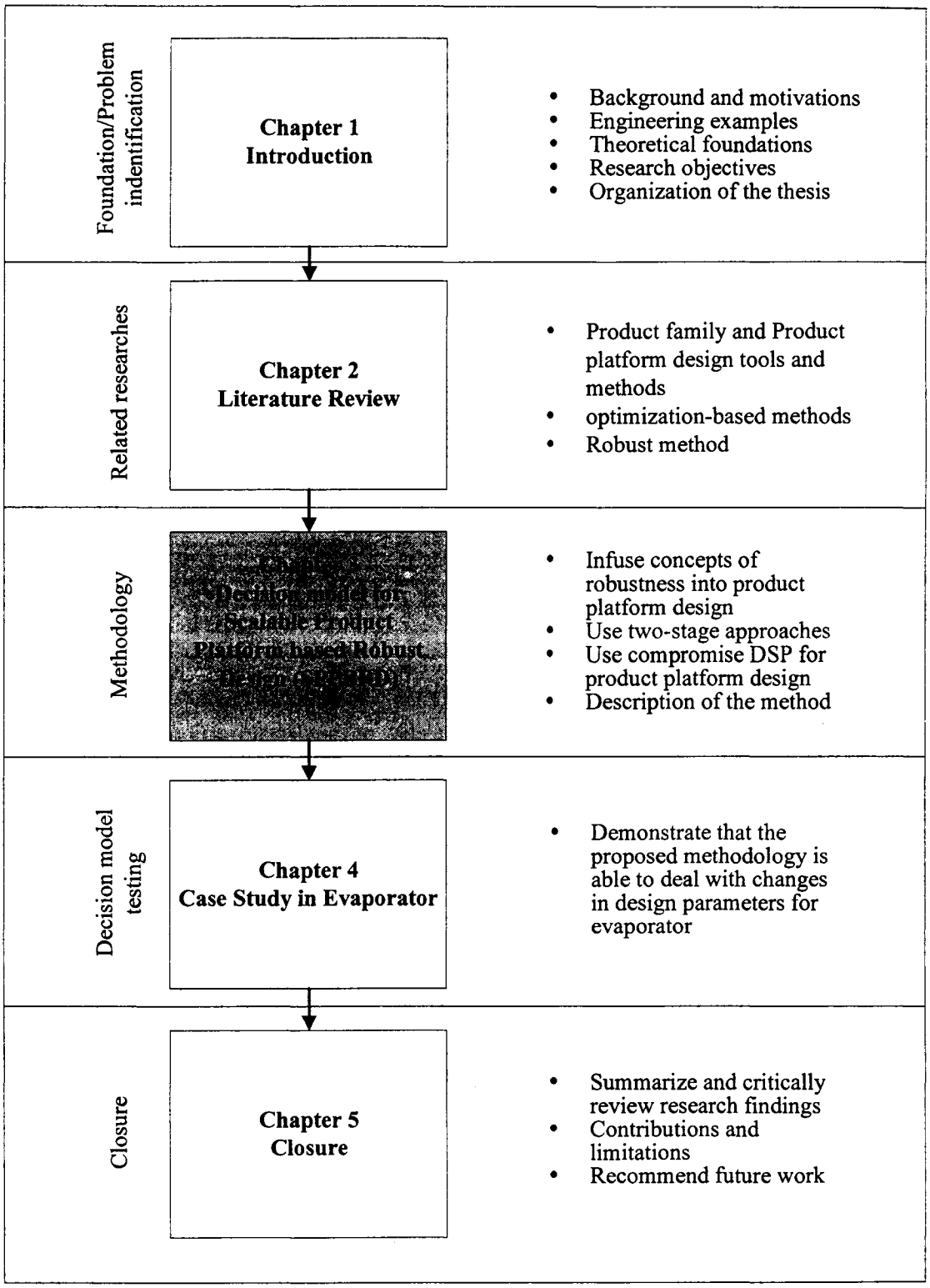

Figure 3.1. Thesis Roadmap

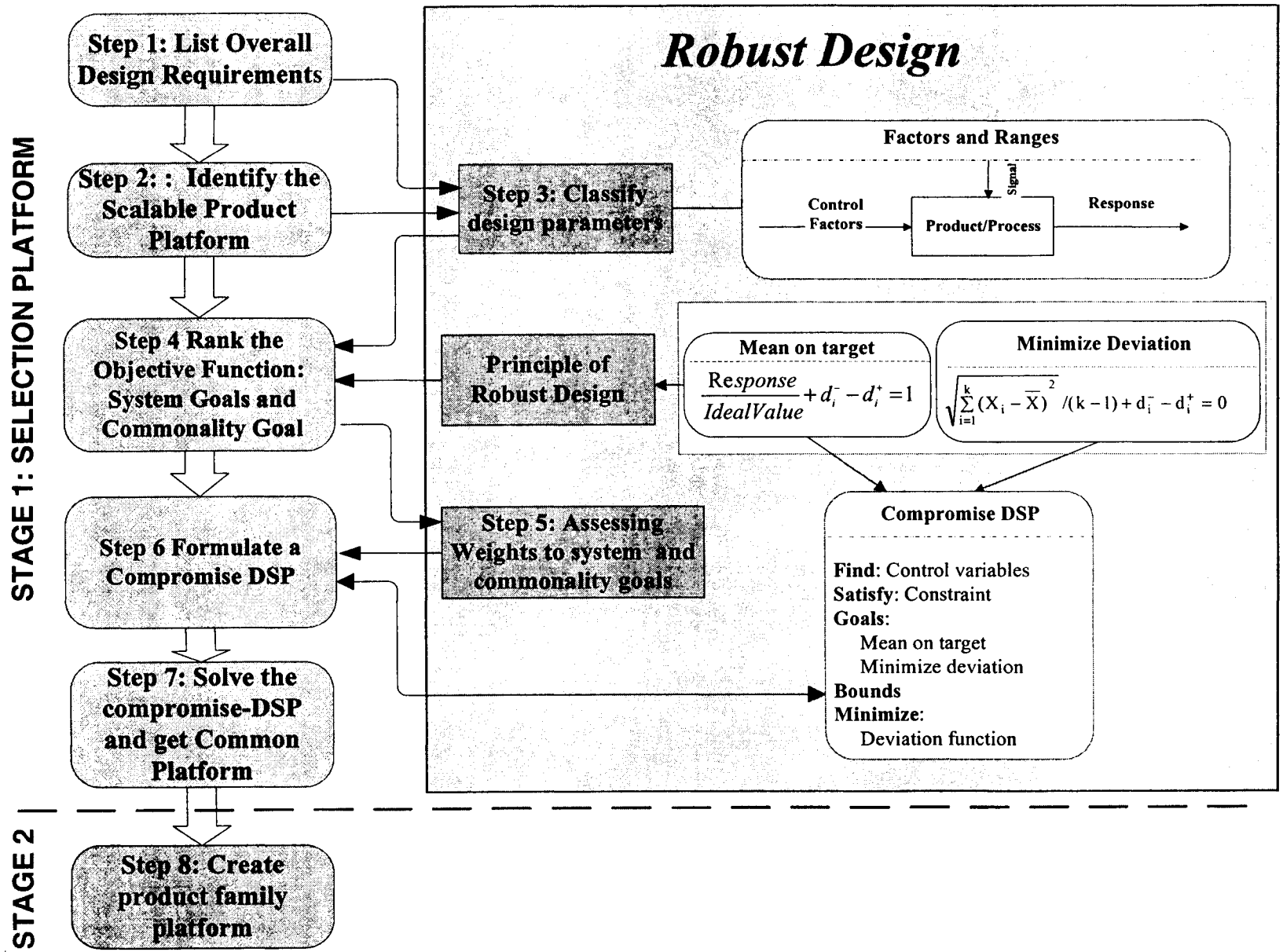

Figure 3.2. Flow Chart of SPPBRD

3.1 Infusion of Robust Design Principles to Scalable Platform Design

In figure 3.2, the contents in the right green square that include step 3, step 5 and two goals about mean on target and minimizing deviation describe how to infuse robust design into the whole model.

In robust design, the process of robust design generally starts with identifying the initial settings of control factors and their ranges, as well as the noise factors (i.e., uncontrollable parameters). The relationship between different types of design parameters or factors can be represented with a P-diagram, where P represents either product or process (Phadke, 1989). The details to classify the design parameters for evaporators will be discussed in step 3.

Generally speaking, the fundamental motive underlying robust design, as originally proposed by Taguchi, is to improve the quality of a product or process by not only striving to achieve performance targets but also by minimizing performance variation, which is the principle of robust design and will be described in section 3.2 step 4. In other words, there are two goals in robust design: one is "strive to achieve performance target", and the other goal is "minimize performance variation". In actual manufacturing, almost every manufacturer wants to get perfect performance, and if cannot be perfect they will try their best to get the desired level, so if the target is the perfect performance, what they should and want to do is to try to let the mean close to the desired performance, which means to achieve the performance target or " moving the mean to the target" and can be described in formula 3.5 in figure 3.8, or can be expressed to the following formula:

$$\frac{Response}{Ideal} + d_i^- - d_i^+ = 1$$

From this formula we can get, when the response (performance) is close to the ideal value (target), the ratio between response and ideal value will be close to 1, and the deviation variables (d_i^- and d_i^+) will be close to zero, which also indicates another goal of the robust design, "minimize the deviation". Additionally, one objective of the product platform is to embrace commonality, and the need for commonality requires a minimum set of design variables whose deviations help satisfy the range of requirements. This brings another goal of robust design: minimize the deviations or minimize performance variation, which is also a named commonality goal and will be introduced and used in the compromise decision support problem in step 4 and step 6.

In practical situations in a framework of product family based robust design, engineers often face multiple quality attributes when designing product or processes to meet various needs of customers. To compromise among quality attributes, assessing a weight for each quality attribute is an inevitable task for this situation. Here, the quality attributes are system goals and commonality goals.

3.2 Two-Stage Approach to Scalable Product Platform Based Robust Design (SPPBRD)

The whole process of building a scalable product platform can be divided into 2 stages: the first stage is selection platform stage that includes step 1 to step 7. Stage 1 is formulated to determine two objectives: which design variables should be selected as the common platform variables, and the optimal values for these variables. Through solving the compromise DSP, the common platform can be decided. Once the common platform

parameters and their values have been determined in the first stage, the values of non-platform variables are sought to best satisfy the functional requirements of the individual products during the second stage of the SPPBRD.

3.2.1 Stage 1: Platform Selection in SPPBRD

Step 1: Specify the Overall Requirements

The *space of customization* is the set of all feasible combinations of values of product specifications that a manufacturing enterprise is willing to satisfy (Hernandez et al., 2002). Consider that there are N independent product requirements X_1, X_2...X_k identified that characterize the customer demands on a product. These requirements help to define the N-dimensional space of customization $M_k = \{X_1, X_2 ... X_k\}$. A space of customization definition involves the following components:

- Identifying which parameters of the product should be varied depending on the needs of the customer
- The range of variety that needs to be offered for each parameter
- The customer demand in the space of customization
- The possible variability in demand in the future

Each dimension of the geometric space represents one of the product parameters in which variety will be offered. The range of each varied parameter determines the bounds of each dimension of the geometric space (Williams, 2004). This step also lists the constraints and bounds of the certain performance or design variable that may come from the customers, the manufacturers, international standards and special industrial standards. From this step we get the overall requirements that are the input to the whole SPPBRD model.

Step 2: Identify Scalable Product Platform

After given the overall design requirements, Step 2 in the SPPBRS is to confirm what kind of platform we should create. In section 1.1.5, the definition of the product platform has been given: the set of parameters (common parameters), features, and/or components that remain constant from product to product within a given product family. From this definition there are two basic different platforms: modular platform and scalable platform. In modular platform the product platform members are instantiated by adding, substituting, and/or removing one or more functional modules from the product platform.

Alternative approach is scalable product platform wherein scalable variables are used to "stretch" or "shrink" the product platform in one or more dimensions to satisfy a variety of market niches (Achelle Messac, 2002).

Before we decide to develop the product platform we should confirm what the market segment the product platform is going to use in. Market segmentation is the process of dividing a total market into market groups consisting of people who have relatively similar product needs; there are clusters of needs. The market segmentation grid provides a link between management, marketing, and engineering design to help identify and map which type of leveraging can be used to meet the overall design requirements and realize a suitable product platform and product family. Here the market segmentation grid serves as an attention-directing tool to help identify potential opportunities for horizontal leveraging, vertical leveraging, or a beachhead approach to product platform design. Market segmentation grid is shown in figure 3.3.

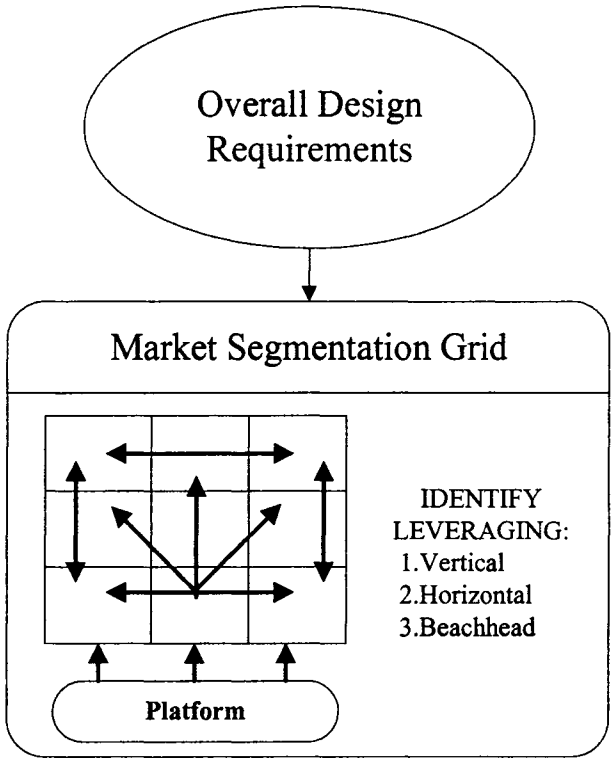

Figure 3.3. Create the Market Segmentation Grid

For the horizontal leveraging strategy (shown in figure 3.5 (b) horizontal), platform subsystems and/or manufacturing processes are horizontally leveraged across different segments, it brings series of related products for different customer groups without having to "reinvent the wheel", and R&D can develop products more rapidly and without less risk (since technology has been proven in other market segments), besides, manufacturing procurement and retooling costs can be minimized. Horizontal leverage generally is used in modular product platform.

For the vertical leveraging, the key platform subsystem and /or manufacturing processes are scaled up or down (shown in figure 3.5 (a) vertical). For the R&D and manufacturing, they almost enjoy the same benefits as horizontal leveraging, besides,

they can leverage knowledge of customer wants and needs within given market segment and the product development is less costly. Vertical leverage is mostly used in scalable product platform.

There have been many researches on the modular product platform that have been introduced in chapter 2, and also many companies have been successful with scalable product platform and corresponding family of products which are scaled around the product platform. However, few people put forward a systematic approach for the scalable product platform based with robust design. Therefore, this thesis focuses on developing a systematic model for the scalable product platform based on robust design.

Step 3: Classify the Design Parameters

The market segmentation grid has been created in last step, in this step, the initial concept exploration space is defined and the problem is formulated as robust design. In the real design situation, we have many design parameters that affect the performances of the products. The process of robust design generally starts with identifying the initial settings of control factors and their ranges, as well as the noise factors (Apichat Sopadang, 2000). Classifying the design parameters is infused in to the SPPBRD that is illustrated in Figure 3.4. Design parameters are grouped as either control factors, response, or noise factors. They can be defined for following as figure 3.4:

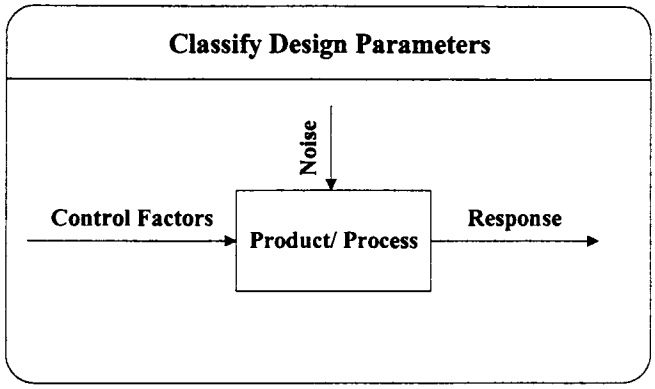

Figure 3.4 Parameter Diagram (Rakesh S. Kulkarni, 2005)

Responses are performance parameters of the system; in the problem formulation, they may be constraints or goals or both and are identified from the overall design requirements and the market segmentation grid.

Control factors are variables that can be freely specified by a designer; settings of the control factors are chosen to minimize the effects of variations in the system while achieving desired performance targets and meeting the necessary constraints. Signal factors also are lumped within control factors because it is often difficult to know, *a priori*, which design variables are control factors and can be used to minimize the sensitivity of the design to noise variations and those that are signal factors and have no influence on the robustness of the system. Control factors represent the to-be-determined design specifications, which describe the characteristics of a design at system level (Apichat Sopadang, 2000).

Noise factors are parameters over which a designer has no control or which are too difficult or expensive to control.

Scale factor is a factor around which a product platform is leveraged either through vertical scaling, horizontal scaling, or a combination of the two. (Timothy W. Simpson, 1998)

The relationship between each type of scale factor and the three types of leveraging are as follows:

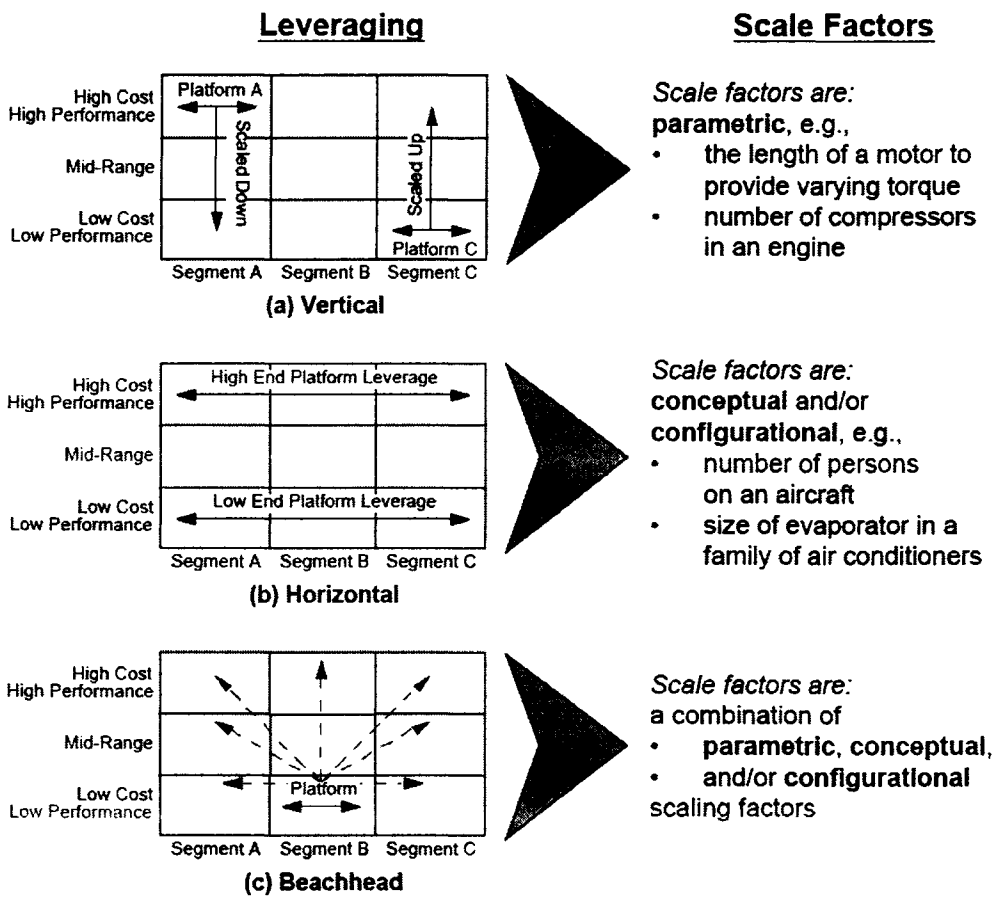

Figure 3.5 Relationships of Scale Factors to the Market Segmentation Grid (Timothy W. Simpson, 1998)

Step 4: Define the Objective Functions based Robust Design

In this step, the robust design integrates the scalable product platform perfectly.

The concept of robust design, which is originally proposed by Taguchi in 1986, is to make a product performance minimally sensitive to the various causes of variations. At that time, Taguchi advocates the use of an inner-array and out-array approach to implement robust design. The inner array consists of orthogonal arrays (OA) that contains the control factor settings; the outer-array consists of the OA that contains the noise factors and their settings that are under investigation. The combination of the inner-array and outer-array constitutes the product array. The product array is used to test various combinations of the control factor settings systematically over all combinations of noise factors after that the mean response and standard deviation may be approximated for each run using the equations:

Response mean: $$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Standard deviation $$S = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \overline{X})^2}{n-1}}$$

Preferred parameter values can be determined through analysis of signal –to-noise (SN) ratio, factor levels that maximize the appropriate SN ratio are optimal. The most useful type of SN ratio is:

$$SN_T = 10\log(\frac{\overline{X}^2}{S^2})$$

There are some criticisms of Taguchi's implementation of robust design through the inner and outer array approach: it requires too many experiments, the analysis is statistically questionable because of the use of orthogonal arrays, it does not

accommodate constraints, and so on. After Taguchi, there are many people improve Taguchi's robust design theory. Currently, the principle of the robust design is to move the mean to target and to minimize the effect of the causes of variation without eliminating the causes. The mean of performance is assumed to be at the mean of the design variables.

Almost all manufacturers or companies' dream is to get the best performance with least cost or most profit. This point can be achieved through implementing robust design's principle above, section 3.1 has narrated how to move the mean to target and what is commonality goal.

Williams (Williams, 2003) infused the utility based on compromise Decision Support Problem in handling multiple objectives in the product platform design.

The multiple objectives in SPPBRD include system goals that may conflict and commonality goal. The system goals can be expressed as achieving performance objectives that close to target and satisfy all requirements, i.e., bring mean on target.

Simpson provided commonality goal in 2002. The need for commonality requires the use of a minimum set of design variables whose deviations help satisfy the range of requirements. Hence, one objective in platform design is to find the smallest set of design variables whose variation will satisfy the range of performance requirements as best as possible. This is accomplished by creating a goal of minimizing the total deviations in as many design variables as possible. This goal is called the commonality goal. These multiple objective will be used as the goals in the compromise decision support problem (c-DSP) model.

An objective function can be formulated by continuous analysis of the space using an integral equation [3.1] (Simpson, 2002), and its mean can be calculated with this method.

$$Y = \int_{X_1,\min}^{X_1,\max} \int_{X_2,\min}^{X_2,\max} \cdots \int_{X_n,\min}^{X_n,\max} Y(X_1, X_2, \ldots, X_n) dX_1 dX_2 \ldots dX_n \qquad [3.1]$$

The standard deviation can be gotten from equation [3.2] (Simpson, 2002),

$$\sigma = \sqrt{(\frac{dy}{dx_1})^2 \sigma_{x1}^2 + (\frac{dy}{dx_2})^2 \sigma_{x2}^2 + \ldots + (\frac{dy}{dx_n})^2 \sigma_{xn}^2} \qquad [3.2]$$

Or the mean of the objective can be formulated by sampling methods using a summation equation 3.3; the deviation can use equation [3.4] (Taguchi, 1986).

$$\overline{X} = \sum_{i=1}^{k} X_i / k \qquad [3.3]$$

$$\sigma^2 = \frac{\sum_{i=1}^{k} [X_i - \overline{X}]^2}{(k-1)} \qquad [3.4]$$

Step 5: Assessing Weights to Quality attributes

In many real-world situations, engineers often face multiple quality attributes when designing products or processes. Assessing a weight for each quality attribute is an inevitable task for this situation. However, classical methods for assessing the weights may not be well suited, particularly when dealing with decision problems associated with fuzziness such as human linguistic preferences. There are two major tools - fuzzy set theory and Monte Carlo simulation to solve this problem. The fuzzy set theory may be a good means for modeling uncertainty or imprecision arising from environment that human beings are heavily involved in the process of decision analysis. Although customers should determine the quality attribute's degrees of importance, the designers'

opinions are also valuable due to their knowledge and experiences. The combination between customers and designers' preference is needed and used in the multi-response optimization model to determine the optimum settings of products and processes.

Apichat Sopadang, Young Jin Kim, and Byung Rae Cho (2001) developed a method for assessing weight of multiple quality attributes. The proposed procedure comprises four steps described below and the procedure can be depicted in Figure 3.6:

1. Transform the linguistic expression into fuzzy number
2. Normalize designer's decision matrix
3. Assign crisp scores to fuzzy number
4. Assign weight for quality attributes

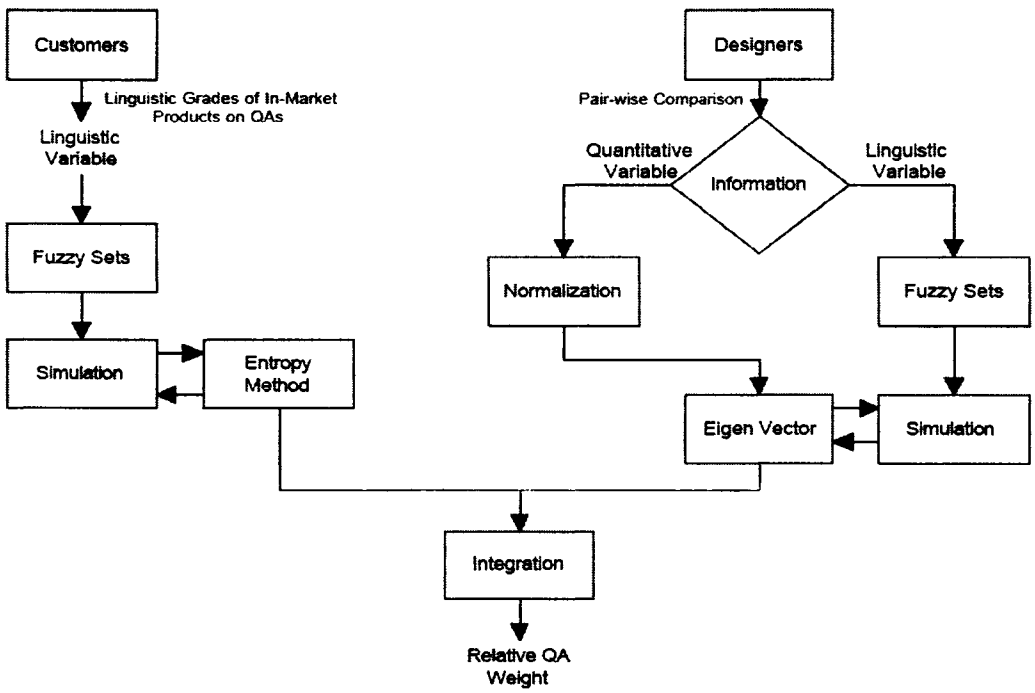

Figure 3.6 the procedure for attribute ranking analysis (Apichat Sopadang, 2000)

Step 6: Formulate a Compromise Decision Support Problem

Once the overall requirements, the market segmentation, factor classification and ranges, and objective function have been decided, the next step is to formulate the multistage compromise decision support problem. It is imperative that product constraints or goals given in the overall design requirements that are not captured within the desired platform leveraging strategy be included in the compromise DSP (Simptson, 1999). It is used to determine the values of the design variables that satisfy a set of constraints and achieve a set of potentially conflicting goals as closely as possible. The compromise DSP is a hybrid formulation based on mathematical programming and goal programming for solving Multi-objective optimization problems (Mistree, 1993).

The basic compromise DSP model can be described as below:

Given: Mathematical relationships, and constants
Find: Control variables
Satisfy:
Constraints
Goals:
System goals
Commonality goals
Bounds
Minimize:
Deviation function

Figure 3.7. Basic compromise-DSP to determine the product platform

According to the outline in above figure, the generalized formulation of the compromise DSP can be stated in words as follows in figure 3.8.

After analyzing the Step 1, Step 2,Step 3 and Step 4, we can get the mathematical relationships, constants and system constraints as given conditions. When designing the

product platform we need to get the values of control variables that are modeled to three variables: the mean μ and the standard deviation of design variables, and the deviation variables. The compromise DSP helps to find these control variables that satisfy the constraints and bounds on the design and achieve as closely as possible the system goals.

The constraints and goals targets are imposed on the mean and standard deviation of the performance so as to satisfy the range of performance requirements for the entire product family. The following detailed descriptions are provided to explain these concepts step by step. As shown in equations in (3.1) through (3.4) in figure 3.8, the constraints for meeting a range of performances are generally classified into 4 categories that include:

a. Equality constraints on performance with different desired values from product to product

b. Equality constraints on performance with the same desired value from product to product

c. Inequality constraints on performance with different limiting values from product to product

d. Inequality constraints on performance with the same limiting value from product to product

For category (a), two sets of constraints are imposed to achieve the mean location and dispersion of the performance modeled (in Eq.3.1). The modeling of category (b) is identical to category (a) but with the dispersion set as zero because the desired values of all the equality constraints are the same in this case (Eq.3.2). For category (c) and (d), only the mean performance is modeled, when the limiting values of all the products are

same in category (d), the worst case among all of the products is identified to satisfy the constraint (Eq.3.4).

The aim of the compromise DSP is to find values for system variables that satisfy the constraints and the bounds on the design and achieve two system goals as closely as possible: moving the mean to target and minimizing the deviation in response. The extent to which each goal is achieved is modeled by the system goals:

$$\text{Response/Ideal value} + d_i^- - d_i^+ = 1$$

$$\text{Ideal value/Variance} + d_i^- - d_i^+ = 1$$

These two equations can be combined into Eq.3.5 in the figure 3.8, in which deviation variables (d_i^- and d_i^+) indicate the extent to which each goal achieves its target value and represents under-achievement or over-achievement of each goal with respect to the target values. The aim of Eq.3.5 is to maximize the value of each individual objective function. A designer would like to achieve the ideal value 1 for each goal, but does not expect to achieve it necessarily. For design requirements that are considered as goals (Eq.3.5), either the goals or the mean of the different goals are modeled. The distribution of the goals is not important because goals represent the designer's wishes, and the targets are used to express the aspiration levels but not necessary the true levels of performance.

On the other side, the need for commonality requires the use of a minimum set of design variables whose deviations help satisfy the range of requirements. So, one objective in the compromise DSP is to find the smallest set of design variables that are accomplished by creating a goal of minimizing the total deviations in as many design variables as possible (modeled in Eq.3.6). This goal is called the commonality goal. Here

the normalizing factors for each performance are assumed as 1 that means the weights for different parameters are same because in the random sampling methods each design parameters has been arranged the same probability.

Given: An alternative that is to be improved

System parameters: $x_k, k = 1,...t$

Mathematical equations of design variables

Constants

n+m number of system constraints

n equality constraints

m inequality constraints

s number of system goals

Weight for the Archimedean case: $w_i, i = 1,...,s$

Find:

Mean of system parameters: $u_{x_k}, k = 1,...,t$

The values of the deviation variables: d_i^-, d_i^+, i=1,...s

Standard deviation of the design variables:

σ_{x_k}, k=1,..,t ,t is the number of design variables for each of the j=1,..,p products

Satisfy:

Equality constraints on performance with a different value for each product of the product platform $A_{ij}(x) = R_{ij}$

This constraint is modeled as $\mu_{A_i} = \mu_{R_i}, and\ \sigma_{A_i} = \sigma_{R_i}$ (3.1)

Equality constraints on performance with the same desired value for each product of the platform $A_{ij}(x) = R_i$

This constraints is modeled as $\mu_{A_i} = R_i$ and $\sigma_{A_i} = 0$ (3.2)

Inequality constraints on performance with different limiting value for each product of the platform $A_{ij}(x) \le R_{ij}$: this constraint is modeled as $\mu_{A_i} \le \mu_{R_i}$ (3.3)

Inequality constraints on performance with the same limiting value for each product of the family $A_{ij}(x) \le R_i$: this constraint is modeled as $A_i(x)_{worst-case} \le R_i$ (3.4)

Goals:

System Goals must be achieve a specified target as far as possible, there is no restriction placed on linearity or convexity.

$$A_i(x)/R_i + d_i^- - d_i^+ = 1, \text{ i=1,...,s} \quad (3.5)$$

The ***commonality goal*** for minimizing the deviation of the system variables, and thus helping in the standardization:

$$(\sigma_{x_1} + \sigma_{x_2} + ... + \sigma_{x_k})/t + d_i^- - d_i^+ = 0 \quad (3.6)$$

Bounds:

$$d_i^-, d_i^+ \ge 0 \text{ and } d_i^- \times d_i^+ = 0 \quad (3.7)$$

$$x_{j\min} \le x_j \le x_{j\max}; j = 1,...p \quad (3.8)$$

Minimize:

The *deviation function* which is a measure of the deviation of the system performance from that implied by the set of goals and their associated priority levels or relative weights:

$$Z = \sum_{i=1}^{s} w_i(d_i^- + d_i^+); \ \sum_{i=1}^{s} w_i = 1; w_i \ge 0 \quad (3.9)$$

Figure 3.8 Compromise DSP for the product platform (modified from Raviraj . Nayak, 2002)

Bounds are specific limits placed on the magnitude of each of the system variable and deviation variables. Each variable has a lower and an upper bound associated with it. Bounds are important for modeling real-world problems because they provide a means to include the experience-based judgment of a designer in the mathematical formulation. In the this thesis, there are two bounds that are modeled in Eq3.7 and Eq.3.8. Eq3.7 means that mathematically a goal is either over-achieved or under-achieved but not both, so one of the deviation variables always must be zero. Eq.3.8 describes every performance or design variable has its design range.

Because there is a tradeoff between achieving commonality within a product family and satisfying the functional requirements of each product, the compromise DSP uses a deviation function which is also called Archimedean formulation and in which weights are assigned to the different goals. The weights for each goal are used to emphasize achievement of one goal more than another and can be calculated with the Step 5. The deviation function is minimized in the solution process and is modeled in equation 3.9.

Compromise DSP has a minimum of two system variables, a graphical representation of a two variable compromise DSP is shown in Figure 3.9.

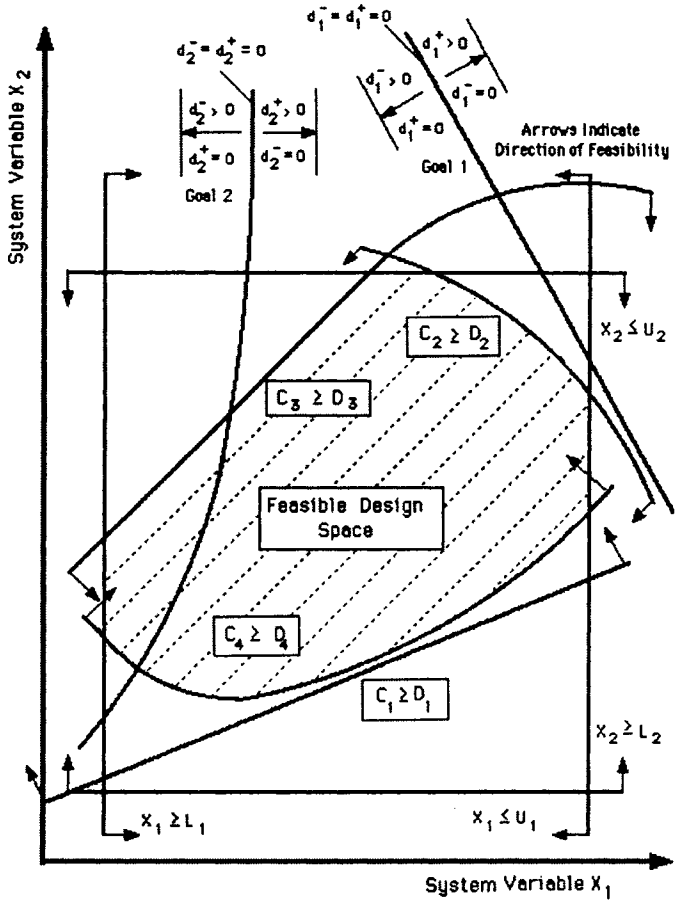

Figure 3.9 Graphical representation of two variables compromise DSP (Mistree, 1993)

Step 7: Solve the compromise DSP and Get Common Platform

To solve the compromise DSP, we can use continuous analysis. We need to express the objectives in terms of the design parameters in continuous analysis, which includes expressing the demand with design parameters. However, the continuous analysis is complex and mathematically demanding because performances' mathematic formulas are very complicated and hard to get mean and deviation; besides, multiple objectives and changing demand are also the reason that continuous analysis is hard to solve the compromise DSP model.

In discrete analysis, the analysis is done on discrete of points in the space that helps to approximate the entire space (Rakesh S. Kulkarni, 2005). A certain resolution is chosen by the designer to discrete the space, thus nodes established in the customization and objective function at every node are calculated. Discrete analysis is the theoretical fundament of sampling methods. For better accuracy, we generally use random sampling methods. For random sampling methods, each sample of the population (the set of individuals, items, or data from which a statistical sample is taken.) has an equal and known chance of being selected. Each sample that is one combination from different design variables represents one possible product in the theoretical product platform. If the entire population will be sufficiently large, then we will get a more efficient result and have a high probability to get the optimal value.

There are some commercial optimization software packages to solve the compromise DSP, such as OptdesX, MATLAB and so on, but for different engineering case, OptdesX and other engineering optimization software also need the users to write some program for specific engineering model with C or FORTRAN. C and FORTRAN are the basic way to solve the problem because C, C++ or FORTRAN is used to develop most of optimization software.

After solving the compromise DSP, we acquire the mean and standard deviations for the system variables. If the standard deviations of the system variables are found to be very small relative to its mean value, it means these variable have very little contribution to achieving the range of performance, and they are then taken as common platform parameters.

3.2.2 Stage 2: Create Product Family Platform

In stage 1 we get the common platform that can be scaled upward, downward or leveraged for variety in product. At the same time, design parameters that have significant variation in the result, cannot be held common for the family and are used as the set of non-platform variables. They are used in the second stage of the SPPBRD to best satisfy the functional requirements of the individual products.

Values of the non-platform variables are sought to best satisfy the functional requirements of the individual products during the second stage of the SPPBRD. One compromise DSP is formulated for each individual product in the family to optimize its non-platform variables. In each of these compromise DSPs, the settings of the common platform parameters identified from the first stage are known. The values of the non-platform design variables (i.e. scaling factors) must be found. The constraints and goals are appropriately modeled to satisfy the functional requirements specified for a particular product in the family. This process is also referred to as the instantiation of the product platform to yield the product family.

The Figure 3.10 shows how the scalable platform can be created. Here, the market segmentation grid can be applied and the product family platform can be created.

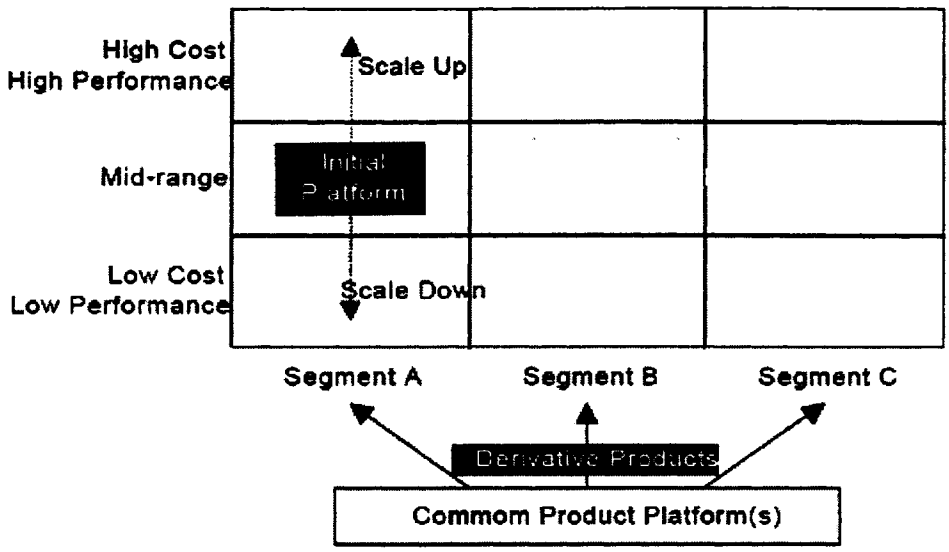

Figure 3.10 Creating product platform

3.3 Summary

The objective of this thesis is to develop a framework for a scalable product platform based robust design model. In chapter 3, a systematic Scalable Product Platform Based Robust Design model was created step by step. The robust concept is infused in the whole development, especially in the compromise decision support problem (c-DSP) to create a decision model for a scalable product platform design, which is also the contribution of this chapter.

CHAPTER 4 DESIGN OF EVAPORATOR PLATFORM

NOTATIONS

A_0	Total heat transfer surface of one-meter longitudinal direction, m^2
A_f	Fin' surface area in one meter tube, m^2
A_l	Area of the narrowest cross-section, m^2
A_r	Area of tube surface in one-meter longitudinal direction, m^2
A_y	Area of air inlet area, m^2
$A_{crosstube}$	The cross section area of tube, m^2
$A_{ij}(x)$	Actual attainment: the *ith* performance of the *jth* product, $i = 1,...,t;\ j = 1,...,p$
b	Fin spacing, distance between the centerline of two adjacent fins
B	Width of the evaporator, m
B_f	The diameter of the hexagon, m
Cost	The total cost of the product platform, $
C_{Al}	Price/kg of aluminum fin, $/kg
C_{casing}	Material cost of casing, $
C_{copper}	Price/kg of tube copper, $/kg
C_{fin}	Material cost of fins, $
$C_{material}$	Cost of raw material, $

C_{Solder}	Approximate cost of the welding material, $/kg
$C_{steelplate}$	Price/kg of steel plate for casing, $/kg
C_{tube}	Material cost of tubes, $
C_{waste}	Cost of waste of material including welding, $
C_{weld}	Welding cost, $
DSP	Decision Support Problem, $/kg
F_0	Area available for heat transfer, m^2
F_{0f}	Outside surface area of the tubes, m^2
F_i	Inside surface area of the tubes, m^2
H	Height of the evaporator, m
H_1	The height of fin, m
IC	Ideal cost or minimum cost of evaporator, $/kg
K_w	Thermal conductivity for tube, $W/m^{2\cdot o}C$
K_f	Thermal conductivity for fin, $W/m^{2\cdot o}C$
K_a	Thermal conductivity for fin, $W/m^{2\cdot o}C$
L	The length of the copper tube, m
N	Refrigerant flow numbers, m
Q_0	Evaporator loads (amount of transferred over time), kW
R_{12}	Refrigerant Freon
$R_{ij}(x)$	Aspiration level: the ith desired performance of the jth product

S_1	Tube spacing, distance between center of circle for two tubes, m
SHR	Sensible heat ratio
SPPBRD	Scalable Product Platform Based Robust Design
V_{ac}	Air velocity of the narrowest cross-section of the evaporator, m/s
V_s	The volume of the welding material, m^3
$\dot{V}$	Air volumetric flow rate, m^3/h
$\dot{W}$	Moisture removal capacity, g/s
Y	Thickness of the evaporator
c_p	Specific heat, $kJ/kg \cdot K$
d_0	Outside tube diameter, m
d_1	Humidity of air inlet of the evaporator, g/kg dry air
d_2	Humidity of air outlet of the evaporator, g/kg dry air
d_i^-	Negative deviation variables, representing under-achievement
d_i^+	Positive deviation variables, representing over-achievement
h	Enthalpy, kJ/kg
m_a	Mass of dry air per hour, kg/h
m_R	Mass flux of refrigerant, kg/s
m_{R0}	Mass flux of refrigerant in each tube, kg/s
p	Number of products in the product platform
$p_{q,b}$	Saturated wet air pressure, Pa

q_i	Heat flow density, kJ/kg
s	Number of system goals
t	Number of design variables of each product
t_0	Vaporization temperature, °C
t_1	Air inlet temperature, °C
t_2	Air outlet temperature, °C
t_f	Average temperature of air, °C
t_k	Condenser temperature, °C
t_ℓ	Dew point temperature, °C
t_{wo}	Temperature of tube surface, °C
v_m	Flow rate of refrigerant, $kg/m^2 \cdot s$
w_i	Weight of system and commonality goals, $i = 1,...,s$
x_k	System desire variables, $, k = 1,...,t$
Δt_0	LMTD between outside surface of tubes and air, °C
Δt_m	LMTD between air and refrigerant, °C
α_i	Refrigerant side heat transfer coefficient, $W/m^2 \cdot °C$
α_{of}	Airside heat transfer coefficient, $W/m^2 \cdot °C$
δ	Thickness of fin, m
η_0	Total fin efficiency

η_f	Fin efficiency
λ	Overall heat transfer coefficient, $W/m^2 \cdot {}^oC$
μ_{A_i}	Mean value of the ith actual performance in the product family
μ_{R_i}	Mean value of the ith desired performance in the product family
μ_{x_k}	Mean of design variables, $k = 1,...,t$
ρ_{copper}	The density of the copper tube, kg/m^3
ρ_{fin}	The density of the aluminum fin, kg/m^3
$\rho_{steelplate}$	The density of the steel plate of casing, kg/m^3
σ_{A_i}	Standard deviation of the ith actual performance
σ_{R_i}	Standard deviation of the ith desired performance
σ_{x_k}	Standard deviation of the design variables
ν	Air kinematic viscosity, m^2/s
v_a	Dry air specific volume, m^3/kg
φ	Relative humidity
φ_1	Relative humidity of air inlet of the evaporator
φ_2	Relative humidity of air outlet of the evaporator

In chapter 1 and 2, the underlying theoretical knowledge that is going to be used in this thesis is presented. In chapter 3, a theoretical model about the scalable product

platform based robust design (SPPBRD) is established. This chapter applies this method to an example to provide empirical and theoretical performance validity of the work.

The main focus of chapter 4 is to answer the question of how the infusion of concepts of robustness into the scalable product platform enables the designer to create platforms that are unaffected by changes in design parameters. Section 4.1 states the problem and requirement of the platform design, section 4.2 narrates technical description of design procedures for individual evaporator design that is the core part and theoretical fundamental of the platform design, and section 4.3 combines the robust design, platform conception and individual evaporator design into the evaporator platform design. Section 4.4 gives the analysis and summary of this case.

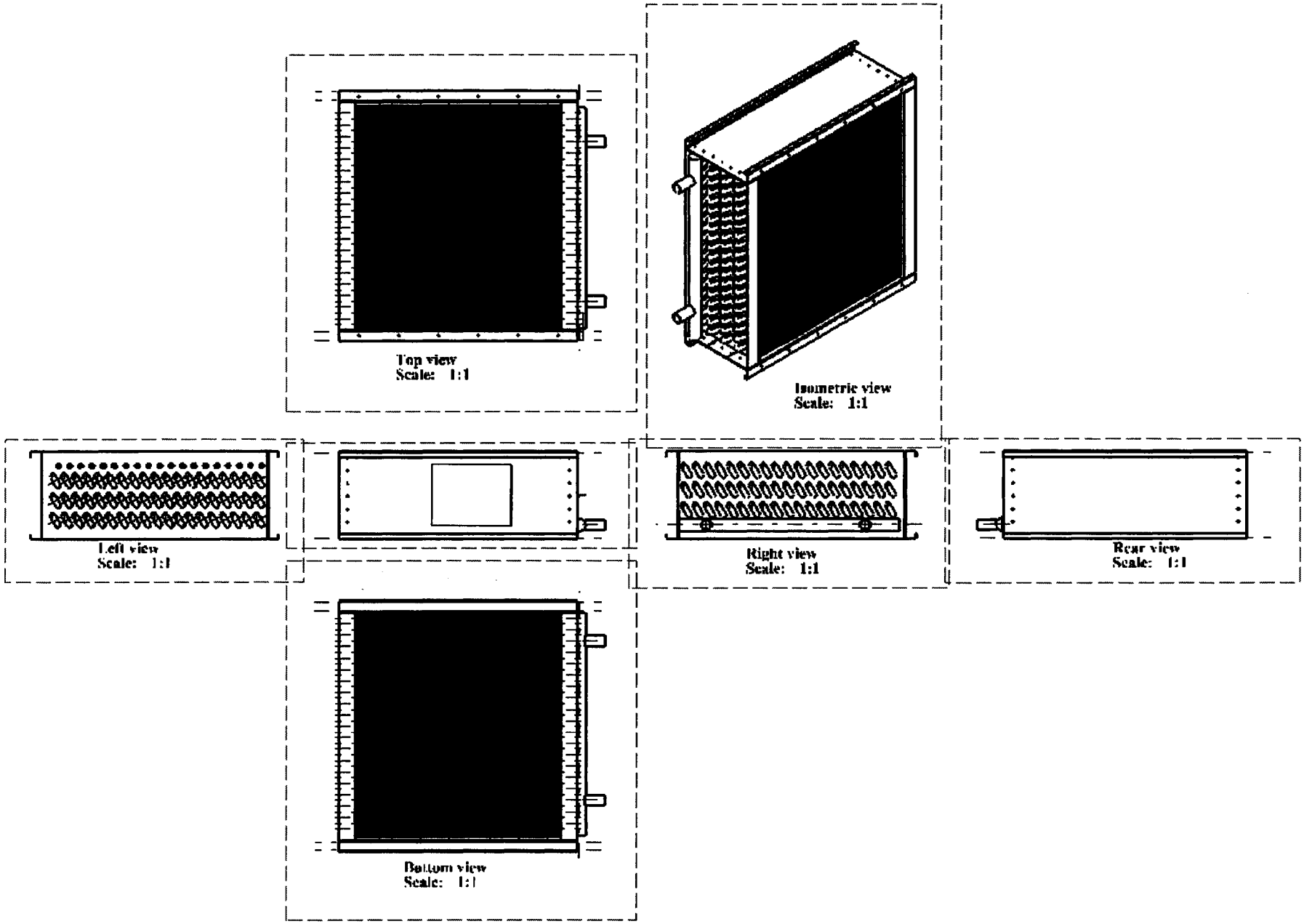

Figure 4.1 All Views of the Evaporator

Figure 4.2 Finned tube evaporator isometric view in Catia

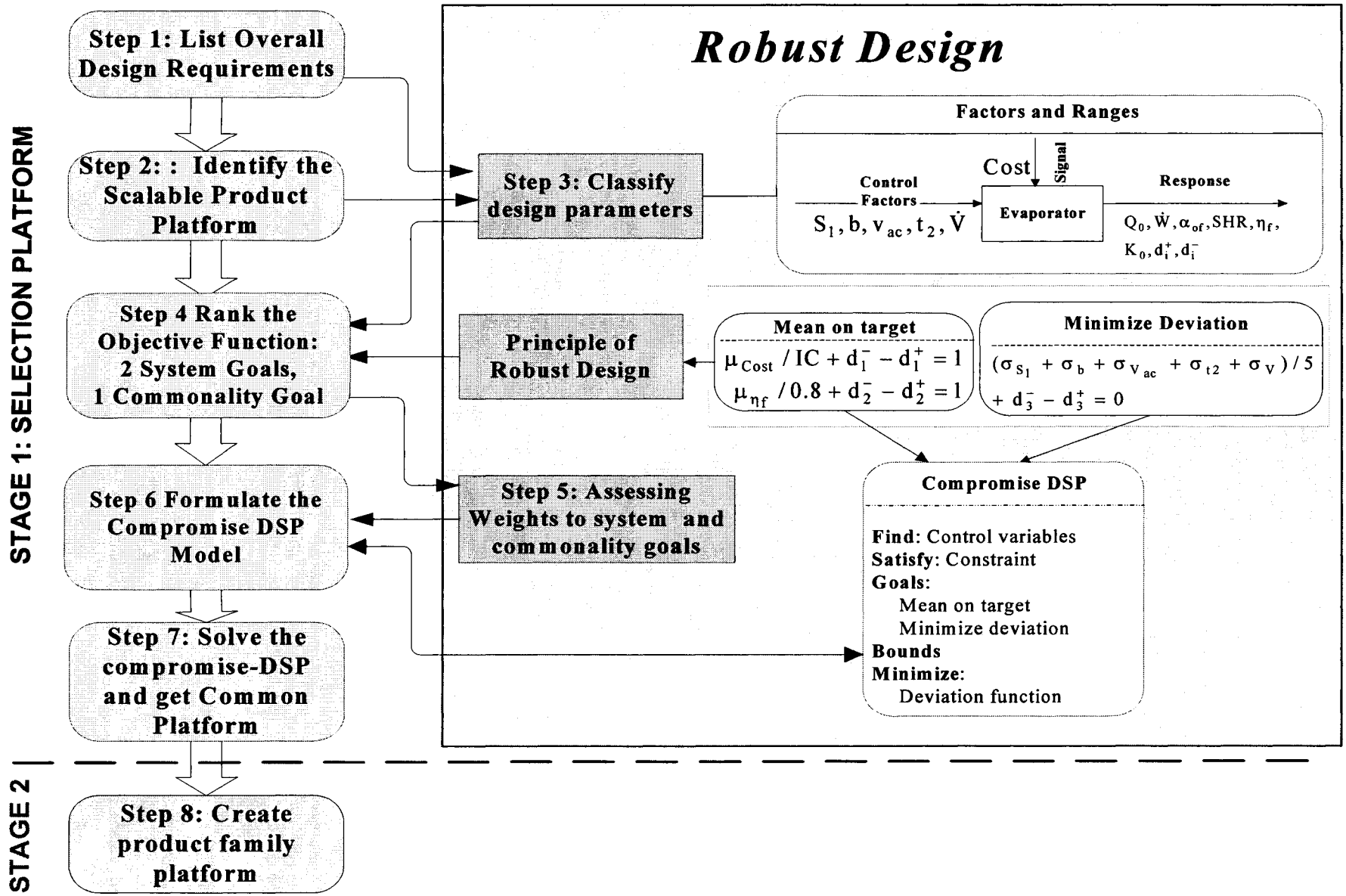

Figure 4.3 Flowchart of the Evaporator Platform bad Robust Design

4.1 Problem Statement

Heat exchangers have many types according to the structure, working principle and application. In the case of this thesis, the product platform for evaporators with a dehumidifying effect will be designed. This is a tube-fin evaporator (i.e. finned tube evaporator, or conventional copper tube-aluminum fin evaporators) and is the key part of the dehumidifier that is one kind of air conditioning equipment.

In this hypothetical case, recently the manufacturer received a big order for a series of evaporators whose structure are similar to the existing products but are ranged by air volumetric flow rate, and the parameter ranges are totally different with the existing evaporators. Another important difference is that evaporators are used in the dehumidifier; they need another function that the existing evaporators don't have---dehumidifying effect. After studying the order, the manufacturer concludes that this order itself could be profitable, since their existing facilities are able to produce the new order, they couldn't need to buy new tools and machines, and the current production is not saturated and has the capacity to accept the new order without planning new facilities layout. In addition, after investigation and study of the market, they also find that there is a potential market for this series of evaporators in the near future. Thus they decide to accept this order.

The next step is to develop the new series of evaporators. Because the existing evaporators failed to embrace commonality, compatibility and standardization among different products and product lines, and their function also has different point with the new order, the plant decides to develop an entire family of evaporators to simplify the finned tube evaporator design process in the future and reduce its variety by standardizing

the evaporators so as to reduce the design cost, as well as manufacturing and inventory cost in the future. At the same time, the plant can shorten the evaporator lead-time, reduce the cost and improve quality to gain more customers and gain a competitive advantage over other leading manufacturers.

One assumption of the new product platform design is that we assume the product platform is adaptable to any of the following changes:

- Changes in the markets including the demand/ order changes
- Changes in technology and/or resources
- Changes in system environments and government legislation (such as refrigerant limit)

According to this order, the design objective of the manufacturer is to develop a brand new robust product platform of ten evaporators (j=1, ..., 10) that satisfies a range of air volumetric flow rates ($\dot{V}$)and will give maximum commonality (for design variables). Details are shown in section 4.3.3 step 1.

4.2 Physical Description and Nomenclature for Evaporator

Dehumidifier is one kind of air conditioner and can remove both sensible heat and latent heat (humidity) by cooling the outside air below the dewpoint to condense out water. It can be used not only for specific application such as precise appliances, special storehouse but also for comfortable air conditioner, to remove moisture. The function of evaporator in dehumidifier is to isolate two different mediums so that they do not touch or mix together, and to transfer heat from refrigerant in tubes to ambient air, and when the surface temperature of the evaporator is lower than the air dewpoint temperature,

some moisture in the air will be condensed to water and drop down to the pan under the evaporator and then flow out of the dehumidifier.

The basic finned tube evaporator consists fins, tubes, U bends, and casing. The all views drawing of the finned tube evaporator is presented in Figure 4.1 and Figure 4.2. To achieve heat transfer, there must be a difference in temperature between the two mediums (air and refrigerant here), a pathway made of materials that allows conduction of heat so it can convey from one location to another, and a means of exposing the heat to the fluid medium. If any of these items are missing, heat transfer will not occur.

4.2.1 Relevant Analyzed for Finned Tube Evaporator

The heat transfer is reflected in the basic relationship from which all heat transfer equations are derived:

$$Q_0 = \lambda F_0 \Delta t_m$$

Changing any one of these values affects the amount of heat that is transferred.

Generally, there are following procedures to design individual evaporator that are also necessary to develop the evaporator platform:

1. Air properties and evaporator loads
2. Evaporator's structure
3. Airside heat transfer coefficient
4. Fin and tube parameters, fin efficiency, temperature difference, heat transfer areas, tube length and tube numbers
5. Refrigerant-side heat transfer coefficient
6. Overall heat transfer coefficient

7. Correct evaporate temperature, heat transfer temperature difference

The following I detail each of above procedure step by step and list the relevant parameters and formulas used to build the evaporator.

1. Air properties and evaporator loads

Figure 4.4 Enthalpy-humidity diagram of air through evaporator shows inlet and outlet air properties and air change process.

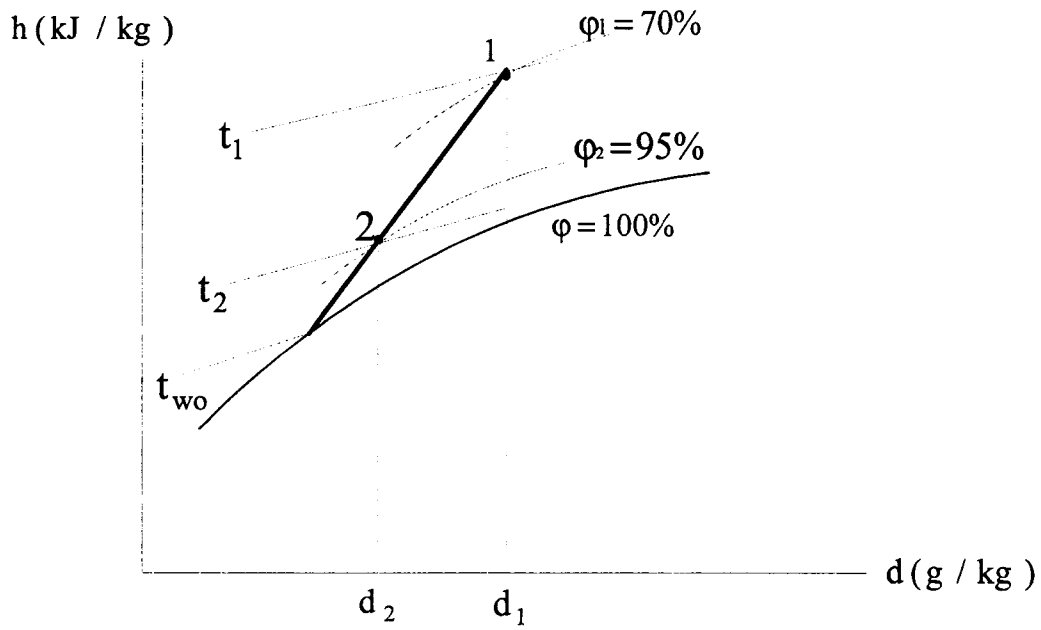

Figure 4.4 Enthalpy-humidity diagram of air through the evaporator

According to the air inlet temperature t_1, relative humidity φ_1, φ_2 we can get humidity d_1, d_2, enthalpy h_1, h_2 for air inlet and outlet point, and dew point temperature t_ℓ from the enthalpy-humidity diagram or from the following mathematical formulas from (4-1) to (4-5) (4-1 to 4-5 were developed by Xue, D.H in 1999; 4-6 to 4-39 were developed by Wu, Y.Z in 2004).

$$d_1 = 0.622\frac{\varphi_1 p_{q,b}}{101325 - \varphi_1 p_{q,b}} \tag{4-1}$$

$$h_1 = 1.01 + 0.001 d_1 (2501 + 1.84 t_1) \tag{4-2}$$

$$T_1 = 273.15 + t_1 (K) \tag{4-3}$$

When $T_1 \geq 473K$, the saturated wet air pressure $p_{q,p}$ comes from:

$$\ln(p_{q,b}) = C_8 / T_1 + C_9 + C_{10} T_1 + C_{11} T_1^2 + C_{12} T_1^3 + C_{13} \ln(T_1)$$
$$C_8 = -5800.2206$$
$$C_9 = 1.391$$
$$C_{10} = -0.04860239 \tag{4-4}$$
$$C_{11} = 0.41765$$
$$C_{12} = -0.14452$$
$$C_{13} = 6.54597$$

The Dew point temperature t_ℓ is developed by:

$$t_\ell = 8.22 + 12.4 \ln(\frac{\varphi_1 p_{q,b}}{1000}) + 1.9[\ln(\frac{\varphi_1 p_{q,b}}{1000})]^2 \tag{4-5}$$

Then the dry air specific volume v_a is:

$$v_a = \frac{RT_1}{101325 - \varphi_1 p_{q,b}} \tag{4-6}$$
$$R = 287.09$$

The mass of dry air for each hour is m_a:

$$m_a = \frac{\dot{V}}{v_a} \tag{4-7}$$

The outlet air properties (point 2 in figure 4.4) when air leaves the evaporator can be got from the same method as point 1. The moisture removal capacity $\dot{W}$ is:

$$\dot{W} = \frac{(d_1 - d_2) m_a}{1000} \tag{4-8}$$

From the above steps, the evaporator loads Q_0 can be calculated from (4-9):

$$Q_0 = m_a(h_1 - h_2)/3600 \text{ (kW)} \quad (4\text{-}9)$$

2. Confirm evaporator's structure

The air's temperature drop in evaporator is not big, so the specific volume has not large changes, and the average air volumetric flow rate can be seen as the air inlet volumetric flow rate. Then the flow area of the narrowest cross-section between two tubes of the evaporator A_1 is:

$$A_1 = \dot{V}/3600V_{ac} \quad (4\text{-}10)$$

Because the limitation of equipment tools and capacity specifications, we choose the copper tube specifications as $\phi 15\times 1$ (outside diameter is 15mm and thickness is 1mm); the arrangement of the tubes is staggered as equilateral triangle shown in Figure 4.5, then the vertical tube spacing is equal to horizontal tube spacing; assume thickness of fin δ is 0.3mm and there are 20 tubes per row. Then air inlet area A_y is given by:

$$\frac{A_1}{A_y} = \frac{(S_1 - d_0)(b-\delta)}{S_1 b} \quad (4\text{-}11)$$

$$A_y = A_1 S_1 b/\{(S_1 - d_0)(b-\delta)\}$$

The height of the evaporator H is:

$$H = S_1 \times 20 \quad (4\text{-}12)$$

The width of evaporator B is given by:

$$B = A_y / H \quad (4\text{-}13)$$

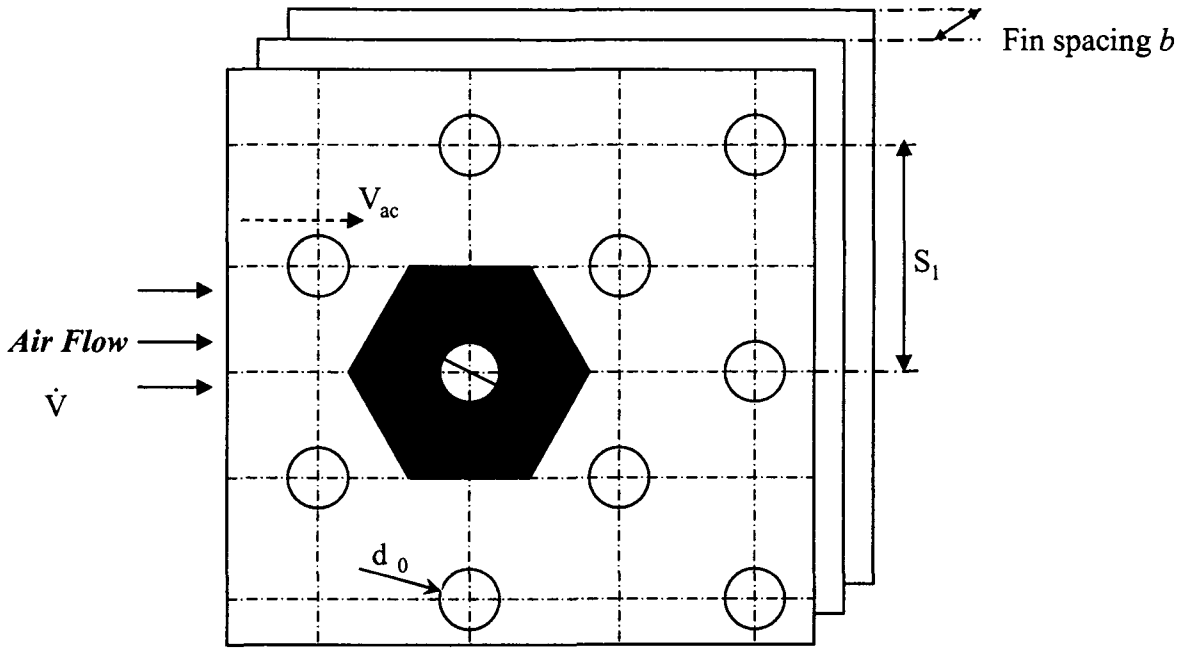

Figure 4.5: Staggered Tube Configuration

3. Calculate airside heat transfer coefficient

The calculating formula for airside heat transfer coefficient α_{of} is given by:

$$\alpha_{of} = 0.205\frac{K_a}{b}\left(\frac{V_{ac}b}{v}\right)^{0.65}\left(\frac{d_0}{b}\right)^{0.4}\left(\frac{H_1}{b}\right)^{-0.14} \qquad (4\text{-}14)$$

The average temperature of air $t_f = (t_1 + t_2)/2$

Through t_f, K_a and v which can be gotten from property handbook, the height of fin H_1 is:

$$H_1 = (B_f - d_0)/2 \qquad (4\text{-}15)$$

The sensible heat ratio (SHR) can be calculated from the following formula:

$$SHR = C_p(t_2 - t_1)/(h_2 - h_1) \qquad (4\text{-}16)$$

$$C_p = 1.0049 + 1.8842\frac{d_2}{1000}(kJ/kg°C) \qquad (4\text{-}17)$$

4. **Fin tube parameters, fin efficiency, temperature difference, heat transfer areas, tube length and tube numbers**

Hexagon fin unilateral area f can be calculated from:

$$f = 6(B_f / 2)(B_f / 2)ctg60^0 - \pi d_0^2 / 4 \qquad (4\text{-}18)$$

Then the surface area of fin in one-meter longitudinal direction A_f is:

$$A_f = f\frac{1000}{b} \qquad (4\text{-}19)$$

Hexagon fin efficiency is given in the following formula:

$$\eta_f = \frac{\tanh(mR_0\zeta')}{mR_0\zeta'} \qquad (4\text{-}20)$$

$$m = \sqrt{\frac{2\alpha_{of}}{SHR \cdot K_f \delta}} \qquad (4\text{-}21)$$

$$K_f = 203.5 W/m \cdot {}^{\circ}C$$

The normalized factor ζ' can be get from the following formula for hexagon fin:

$$\zeta' = \{(B_f / d_0) - 1\}\{1 + 0.35\ln(B_f / d_0)\} \qquad (4\text{-}22)$$

$$R_0 = 0.5d_0$$

Air cooling process of evaporator is shown in the enthalpy-humidity diagram Figure 4-4, extend the line (from point 1 to point 2) to saturated curve (relative humidity 100%), and from the point of intersection, we get the average temperature of the outside of tubes t_{wo}, then get the log mean temperature difference (LMTD) Δt_0:

$$\Delta t_0 = \frac{t_1 - t_2}{Ln\frac{t_1 - t_{wo}}{t_2 - t_{wo}}} \qquad (4\text{-}23)$$

According to equation $Q_0 = \lambda F_0 \Delta t_m$ and thinking about resolve water and icing of the tube, the outside surface area of the tube is given by:

$$F_{of} = \frac{Q_0 \, SHR}{\alpha_{of} \Delta t_0 \eta_0} \tag{4-24}$$

Inside η_0 is given by:

$$\eta_0 = \frac{A_r + \eta_f A_f}{A_r + A_f}$$
$$A_r = \pi d_0 (1 - 1000/b) \tag{4-25}$$
$$A_f = 2f1000/b$$

f has been decided in (4-18), then the total length of tubes is given by:

$$L = \frac{F_{of}}{A_r + A_f} \tag{4-26}$$

The required number of tubes N can be calculated from:

$$N = L/B$$

The N must be integer and thinking about the allowance of the evaporator loads we add extra 10% of tube length for the evaporator, then:

$$N=[(L\times 1.1)/B] \tag{4-27}$$

The actual heat transfer area is

$$F_{of} = N \times B(A_f + A_r) \tag{4-28}$$

Outside surface area of tube F_r is

$$F_r = N \times B \times A_r \tag{4-29}$$

Inside surface area of tube F_i

$$F_i = N \times B \times A_i$$
$$A_i = \pi(d_0 - 0.002)(1 - 0.0003/b) \quad (4\text{-}30)$$

Total fin surface area F_f is:

$$F_f = N \times B \times A_f \quad (4\text{-}31)$$

5. Refrigerant side heat transfer coefficient

It is supposed that in this product platform the temperature of refrigerant before capillary tube or expansion valve is $35^{\circ}C$, degree of superheat of the refrigerant in evaporator is $5^{\circ}C$, and the vaporization temperature is $3^{\circ}C$. Then the evaporator capacity for 1 kg refrigerant is 121.7kJ/kg (got from Pressure-Enthalpy diagram), and related mass flux of refrigerant m_R will be:

$$m_R = \frac{Q_0}{q_0} \quad (4\text{-}32)$$

The mass flux for each tube m_{R0}(we have assumed there are 20 tubes in each row) is:

$$m_{R0} = \frac{Q_0}{q_0 \times 20} \quad (4\text{-}33)$$

The flow rate of inside refrigerant

$$v_m = \frac{m_{R0}}{0.25\pi(d_0 - 0.002)^2} \quad (4\text{-}34)$$

The heat flow density will be

$$q_i = \frac{Q_0}{F_i} \quad (4\text{-}35)$$

If the heat flow density is larger than $4000\,w/m^2$, using the flowing formula to calculate the inside heat transfer coefficient:

$$\alpha_i = 57.8 \times 0.0199 \times \frac{v_m^{0.2}}{(d_0 - 0.002)^2} q_i^{0.6} \qquad (4\text{-}36)$$

6. **Calculate the overall heat transfer coefficient**

The overall heat transfer coefficient can be calculated from the following formula:

$$\lambda = \frac{1}{(\frac{1}{\alpha_i} + \frac{\delta}{K_w})(\frac{F_0}{F_i}) + \frac{SHR}{\alpha_{of}} \frac{F_{of}}{F_r + \eta_f F_f}} \qquad (4\text{-}37)$$

$K_w = 383.8 W/m \cdot ^\circ C$

7. **Correct vaporization temperature, temperature difference between refrigerant and air**

After getting heat loads and overall heat transfer coefficient, the temperature difference can be got from:

$$\Delta t_m = \frac{Q_0}{F_{of} \lambda} \qquad (4\text{-}38)$$

Δt_m also can be got from the log mean temperature difference:

$$\Delta t_m = \frac{t_1 - t_2}{\ln \frac{t_1 - t_0}{t_2 - t_0}} \qquad (4\text{-}39)$$

From equations 4-38 and 4-39 we can get the vaporization temperature t_0. Then compare it with that we have assumed in step 6. If the difference is less than 10%, we don't need to calculate again, if not, we need to assume a new vaporization temperature and calculate step 6, step 7 and step 8 again until the satisfied results are found.

4.3 Scalable Finned Tube Evaporator Platform based Robust Design

This section helps to explain the steps in the SPPBRD (Figure 4.3) by applying it to the finned tube evaporator platform design.

4.3.1 Infusion of Robust Design Principles to Scalable Platform Design

Generally speaking, the fundamental motive underlying robust design, as originally proposed by Taguchi, is to improve the quality of a product or process by not only striving to achieve performance targets but also by minimizing performance variation. In another word, there is two goals in roust design, one is "moving the mean to target", and the other goal is "minimizing variation", these two goals will be discussed in section 4.3.3 step 4.

In robust design, the relationship between different types of design parameters or factors can be represented with a P-diagram, where P represents either product or process (Phadke, 1989). The details to classify the design parameters for evaporator will be discussed in section 4.3.3 step 3.

In practical situations in a framework of product family based robust design, engineers often face multiple quality attributes when designing product or processes to meet various need of customers. To compromise among quality attributes, assessing a weight for each quality attribute is an inevitable task for this situation. The quality attributes in this thesis are system goals and commonality goals. In this thesis, because we have assumed designing the platform is not from the beginning of the embodiment phase of the evaporator and many parameters have been decided, we won't assess the

weights for attributes according to the methods introduced in chapter 3. The detailed method will be given in section 4.3.3 step 5.

In the figure 4.3, contents in the right square that includes step 3, step 5 and two goals about mean on target and minimizing deviation describe the robust design.

4.3.2 Two-Stage Approaches to Evaporator Platform based Robust Design

In the whole process of building scalable product platform for the evaporator, there are 2 stages, the first stage is platform selection stage that includes from step 1 to step 7. Stage 1 is formulated to determine two objectives: which design variables should be selected as the common platform variables and the optimal values for these variables. Through solving the compromise DSP, the common platform can be decided. Once the common platform parameters and their values have been determined in the first stage, the values of non-platform variables are sought to best satisfy the functional requirements of the individual products during the second stage of the SPPBRD.

4.3.3 Stage 1: Platform Selection in SPPBRD

Step 1: Specify Overall Requirements

The space of customization is the set of all feasible combinations of values of product specifications that a manufacturing enterprise is willing to satisfy (Hernandez et al., 2002). In the case the space of customization is the set of all feasible combinations of values of evaporator specifications: Air volumetric flow rate, Moisture removal capacity, Relative humidity, Air inlet and outlet temperature, Mass of dry air per hour, Evaporator loads, Height of the evaporator, Width of the evaporator, all these specifications are

varied depending on the needs of the customers. The following are the constraints and bounds of the specifications and performances:

1. The desired air volumetric flow rate ($\dot{V}$) requirement for the ten evaporators are given by the set: {1000,1500,1900,2400,3000,3400,3800,4200,4600,5000} m^3/h
2. The air outlet temperature ranges: t_2 from 8 to 17 °C
3. For proper comparison of the plain fin optimum evaporator designs to the optimum interrupted fin evaporator design, the restrictions for the tube spacing and fin spacing used are (Susan White Stewart, 2003):

   $$30 \text{ mm} < S_1 < 60 \text{ mm}$$

   $$3 \text{ mm} < b < 6 \text{ mm}$$

4. Air velocity of the narrowest cross-section of the evaporator (Wu, Y.Z, 2004):

   $$3 \text{ m/s} \leq V_{ac} \leq 6 \text{ m/s}$$

5. Fin efficiency η_f (Wu, Y.Z, 2004): $0.7 \leq \eta_f \leq 0.8$

Some design specifications applicable to this evaporator platform are given as (some are customer's requirements): the air inlet temperature t_1 is $22 \pm 1^\circ C$, relative humidity is $60 \pm 10\%$, air pressure is 101.3kPa, refrigerant is R12, and condensate temperature is $t_k = 35^\circ C$, temperature of refrigerant before capillary tube or expansion valve is $35^\circ C$, degree of superheat of the refrigerant in evaporator is $5^\circ C$, and the evaporate temperature is $3^\circ C$.

If the customer's requirements change, these design specifications can be changed to other parameters as design variables, and it can be achieved through changing the input of design variables in the software.

Step 2: Identify Market Segmentation for Product Platform

Before a firm starts to develop new products, the first step is to confirm the market targets and identify what kind of market their products will service.

With a given set of performance requirements and the model derived in Section 4.3.3 step 1, we have known customer requirements are just ranged from the volumetric flow rate and air outlet temperature, they have no new requirements about individual evaporator's function, from which we can get that the all evaporators in the platform have the same performance that can be scaled up and down, and they have same construction, same components and same materials. A market segment consists of individuals, groups or organizations with one or more characteristics that cause them to have relatively similar product needs (http://www.udel.edu/alex/chapt9.html, Oct.3, 2005). So in the market segmentation grid, the evaporator can be scaled down like segment A or scaled up like segment C in figure 4.6. From the low performance and low cost to high performance and high cost, the design parameters are scaled up and performances are still same. So the product platform for this case is scalable product platform.

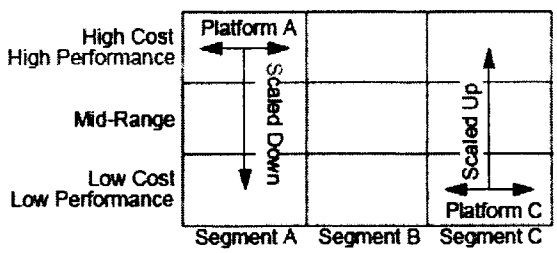

Figure 4.6. Evaporator Market Segmentation Grids

The market segmentation grid shown in Figure 4.6 depicts the desired leveraging strategy for this evaporator example. The goal is to design an evaporator platform, which can be leveraged vertically for different market segments that are defined by the volumetric flow rate needs of each market. In this specific example, ten instantiations of the evaporator are to be considered; moreover, in order to reduce cost, size, it is supposed the best evaporator is the one that satisfies its performance requirements with the least overall cost and greatest efficiency.

Step 3: Classify the Design Parameters

In this real design situation, we have many design parameters that affect the performance of the evaporators. The problem of this step is formulated as a robust design. The purpose of this step is to identify and classify different design parameters that are control factors, noise factors and responses.

Control factors represent the to-be-determined design specifications, which describe the characteristics of a design at system level (Apichat Sopadang, 2000). In this case, because the customers' requirements, some design standards and equipment constraints, some parameters have been fixed in advance, such as air inlet outlet temperatures (air inlet temperature is 22°C), condensate temperature (35°C), air pressure (1.013bar), tube diameter and thickness (Ø15×1), tubes arrangement (staggered, equilateral triangle), fin thickness (0.3mm), tubes in each row (20 tubes per row), indoor relative humidity ($60 \pm 10\%$). Then in this case, the control factors are: air volumetric flow rate $\dot{V}$, air velocity through the narrowest cross-section V_{ac}, , fin spacing b, vertical and horizontal tube spacing S_1, and air outlet temperature t_2, total length of tube L and

width of the evaporator B. Inside these control factors some may be trivial to control factors that can become held-constant factors.

Responses relate to the overall design requirements or quality attributes. Quality attributes are defined as the product characteristics discernible to consumer and design factors (the physical dimensions that the designer can control and specify). The response factors here are evaporator loads Q_0, moisture removal capacity $\dot{W}$, airside heat transfer coefficient α_{of}, sensible heat ratio SHR, fin efficiency η_f and total heat transfer coefficient K_0.

This step begins from the formulation of problem by classifying the design factors. The classification is illustrated in figure 4.7. Here, one of signal factor is cost.

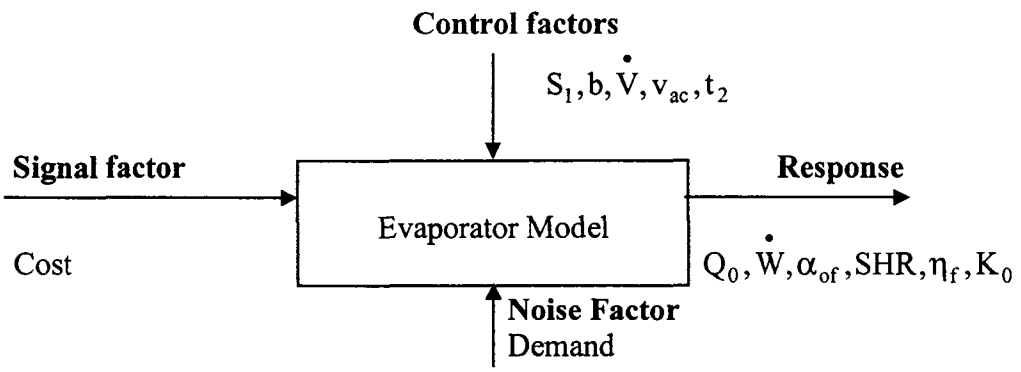

Figure 4.7. P-diagram for Evaporator Platform Design

Step 4: Rank the Objective Functions

The manufacturer has two conflicting goals when designs the SPPBRD---one is to minimize the cost and the other is to maximize the fin efficiency. These two goals

actually narrate one dream that almost every manufacturer wishes to get--- "*move the mean to target*", which also means to get the ideal performance targets for their goals.

Another dream for any manufacturer that is also another robust design principles is to *minimize performance variation*, which can be realized through the *commonality goal.*

The cost can be calculated by dividing the total cost into 6 components: material cost, welding cost, order cost, equipment cost, labor cost and plant cost. Here the labor, order cost and plant cost are ignored because that will relate to very wide issues that are not related to this thesis' objective.

The material cost is determined by the amount of material that is going to be needed to build the evaporator. This cost is comprised of two parts: the cost of the material used in evaporator and the cost of the material wasted by welding and cutting the raw tubes, fins and steel plates to the required dimensions. There are many factors that can affect the waste cost, such as: technology skills, labor skills, tool equipment specifications, and production lot. According to the experience, C_{waste} is arranged as 8% of the raw material $C_{material}$.

The cost of material is therefore given by:

$$C_{material} = C_{fin} + C_{tube} + C_{casing} \qquad (4\text{-}40)$$

The cost of all fins is:

$$C_{fin} = C_{Al}(S_1 20)\{[N/20]S_1(tg60^\circ C)/2\}(B/b) \qquad (4\text{-}41)$$

The cost of the tubes in one evaporator can be given by:

$$C_{tube} = C_{copper}\,\rho_{copper}\,A_{crosstube}\,N \cdot B \qquad (4\text{-}42)$$

Where $A_{crosstube}$ is the area of tube cross section which can be given by:

$$A_{crosstube} = \pi[d_0^2 - (d_0 - 0.001)^2]/4 \qquad (4\text{-}43)$$

In formulas (4-13) and (4-27) N and B have been calculated.

The cost of casing is easy to calculate after the width and height of the evaporator have been decided. In this example we choose galvanized (zinc-coated) carbon steel sheet with thickness 1.2mm as raw material, and give extra more 70mm for the width and height of the casing as allowance to the sheet metal when folding it into desired fabrication to make the casing, so the cost of casing is given by:

$$C_{casing} = C_{sheetmetal}\,\rho_{sheetmetal}\,A_{sheetmetal} \times 0.0012 \qquad (4\text{-}44)$$

$$A_{sheetmetal} = 2\{(S_1 20 + 0.07)([N/20]S_1\sqrt{3}/2 + 0.07) + (B + 0.04)([N/20]S_1\sqrt{3}/2 + 0.07)\}$$

The weld for the tubes and bends takes hand welding. All tube joints should be carefully joined by TIG welding. The welding cost is just composed of the circumferential weld that is given as:

$$C_{weld} = V_s \rho_{solder} C_{solder} \{[L \times 1.1/B]\} \qquad (4\text{-}45)$$

Where the volume of the welding material, V_s, is given by

$$V_s = 4\pi^2(\delta/\cos 30^\circ)^2(60/360)(\frac{d_0}{2} - \delta) = \frac{8}{9}\pi^2\delta^2(\frac{d_0}{2} - \delta)$$
$$= 57mm^3$$

C_{solder} is the approximate cost of welding material ($15/kg hand welded).

There is also a cost associated with ordering the material, C_{order}. Each time the order for raw material is placed, a fee of $250 is assessed in order to cover shipping, handling and stocking in inventory. The cost is based on the number of different sized

metal sheets, fin rolls and tubes of raw material in order; it is not related to the quantity of sheets ordered. However, here the cost is calculated by single product and doesn't consider the order demand and production capacity, so the order cost isn't considered in this case.

The cost of purchasing manufacturing equipment, namely the press machine for casing, fin and bending machine for tubes can be ignored because for every product the equipment cost is almost same and can be looked as a constant.

According to above analysis, the total cost for this evaporator platform can be expressed in following formula:

$$\text{Cost} = 1.08C_{\text{material}} + C_{\text{weld}} \qquad (4\text{-}46)$$

The two objectives minimizing the cost and maximizing fin efficiency are conflicting and are the customers' wants and needs (the formula of fin efficiency has been given in equation (4-20)).

The commonality goal is to minimize the normalized standard deviation of the system variables, in the evaporator platform design, the system variables in the commonality goals are: Air volumetric flow rate ($\dot{V}$), air outlet temperature (t_2), air velocity of the narrowest cross-section (V_{ac}), the tube spacing (S_1), and fin pith (b).

The mean of the performance is assumed to be at the mean of the design variables. There are a few methods to get the standard deviation:

For the discrete random variables, the mean is estimated by:

$$\overline{X}(u) = \sum_{i=1}^{n} X_i / n \qquad (4\text{-}47)$$

The sample variance is:

$$\sigma^2 = \frac{\sum_{i=1}^{n}[X_i - \overline{X}(u)]^2}{(n-1)} \qquad (4\text{-}48)$$

For continuous random variables, the standard deviation can be gotten with first-order Taylor series expansion. In this case, standard deviation is calculated from the formula above. Step 7 in this section will discuss how to solve this problem.

Step 5: Assessing Weights to Quality attributes

As stated in section 3.2.1 step 5, there are some researches about how to assess weights to different attributes. Generally, customer opinion is expressed in subjective and normal terms. In five-point scheme, these may include terms such as: excellent, good, fair, poor, and terrible. In contrast, from the designer's viewpoint, the quality attribute ranking should be independent of product alternatives. Designers compare every attribute with one another to evaluate their relative importance as a pair-wise comparison. However, in this case, because we lack the statistical data from customers and experiment data from the three goals, the goals on cost, fin efficiency, and commonality are assigned equal weights.

Step 6: Formulate a Compromise Decision Support Problem

The core problem of this case is to get the design parameters of the evaporator platform that satisfy a set of constraints and bounds and can achieve the three conflicting goals as well as possible. I create a compromise DSP model to solve this problem in this step.

The compromise DSP is a multi-objective decision model that is a hybrid formulation based on Mathematical Programming and Goal Programming (Mistree,

1993). It is used to determine the values of design variables that satisfy a set of constraints and achieve a set of conflicting goals as well as possible. In a compromise DSP, the objective is to minimize the deviation function, which is a function of the goal deviation variables.

The overall requirements, market segmentation, factor classification and ranges, the mathematical formula of the evaporator have been given from section 4.3.3 step 1 to step 5.

The design objective is to develop a family of ten evaporators to satisfy a range of volumetric flow rate ($\dot{V}$) and air output temperature and other constraints. The constraints and bounds are presented in section 4.3.3 step 1. Each evaporator has 6 design variables (those are shown in commonality goal) that need to be determined during the design process to satisfy the needs and requirements of the product. Because in this thesis the product platform design is not entirely started from except design phase and the beginning of the embodiment phase of the heat exchanger, some parameters have been set, tubes are staggered as equilateral triangle arrange, and other settings such as the tube and bend's diameter and thickness, fin thickness, , inlet air parameters, relative humidity, condenser temperature, and tube number in every row of the evaporator.

There are three goals that need to be satisfied to get a robust design; a decision that gives the best possible combination of the two system goals and one commonality goal. The compromise DSP is used to formulate this problem.

In the formulation of the compromise DSP, the system goals are measured in terms of the deviation of the objective function from the ideal value (*IC*); in this case, the

first objective is to minimize the cost and the second objective is to maximize the fin efficiency. Each set of system objective functions is represented respectively by:

$$\mu_{Cost} / IC + d_1^- - d_1^+ = 1$$

$$\mu_{\eta_f} / 0.8 + d_2^- - d_2^+ = 1$$

The commonality goal is to minimize the standard deviation of the system design variables with a target of zero and give an indication of whether the system variables can be held constant or not within the product family. It can be represented as:

$$(\sigma_{S_1} + \sigma_b + \sigma_{\dot{V}} + \sigma_{t_2} + \sigma_{V_{ac}}) / 5 + d_3^- - d_3^+ = 0$$

The goals on minimizing cost, maximizing fin efficiency and commonality goal can be assigned different weights in section 4.3.3 step 5, the weights can be varied suitably to represent designer preference and the customer's demand for the variety and standardization requirements. The resulting product family is different when the weights assigned to the different goals are changed. The deviation variables are thus found; using the weights of each objective discussed above along with the deviation variables forms the deviation function.

$$Z = \sum_{i=1}^{3} w_i (d_i^- + d_i^+); \quad \sum_{i=1}^{3} w_i = 1; w_i \geq 0$$

The compromise DSP formulation for designing the evaporator is shown in Figure 4.8.

Given:

Evaporator's mathematical models for design variables (see Section 4.2)

Find:

The mean and standard deviation of the design variables ($S_1, b, V_{ac}, t_2, \dot{V}$, L, B, Y), and deviation variables $d_1^-, d_1^+; d_2^-, d_2^+; d_3^-, d_3^+$

Satisfy

Equality constraints : $\mu_{\dot{V}} = 3080 m^3/h$, and $\sigma_{\dot{V}} = 452 m^3/h$

Inequality constraint on fin efficiency and air outlet temperature:

$$0.7 \le \eta_f \le 0.8$$

$$8 \le t_2 \le 17 (^\circ C)$$

System goals that must achieve a specified target as far as possible;

$$\mu_{Cost} / IC + d_1^- - d_1^+ = 1;$$

$$\mu_{\eta_f} / 0.8 + d_2^- - d_2^+ = 1$$

Commonality goal:

$$(\sigma_{\dot{V}} + \sigma_{t_2} + \sigma_{V_{ac}} + \sigma_{S_1} + \sigma_b)/5 + d_3^- - d_3^+ = 0$$

Bounds:

$$d_1^- \cdot d_1^+ = 0; d_1^-, d_1^+ \ge 0; d_2^- \cdot d_2^+ = 0; d_2^-, d_2^+ \ge 0; d_3^- \cdot d_3^+ = 0; d_3^-, d_3^+ \ge 0;$$

$$3mm \le b \le 6mm$$

$$30mm \le S_1 \le 60mm$$

$$3 \text{ m/s} \le V_{ac} \le 6 \text{ m/s}$$

Minimize

$$Z = \sum_{i=1}^{3} w_i (d_i^- + d_i^+); \ \sum_{i=1}^{3} w_i = 1; w_i \ge 0$$

Figure 4.8 Compromise DSP for determining the evaporator family platform

Step 7: Solve the Compromise Decision Support Problem

To tackle the compromise DSP, there are various ways. The first one is continuous analysis, in this way we need to express the objectives in terms of the design parameters. Theoretically, it is feasible, acceptable and accurate. However, when getting the mean and deviation of design parameters, the continuous analysis becomes complex and mathematically demanding due to the consideration of multiple objectives, changing demand and complicated expresses of the functions.

To solve this problem, the standard deviation of the performance can be calculated using first-order Taylor series expansion assuming that deviation is small. For better accuracy, we could use random sampling methods. It should be noted that both of the Taylor series approximation and sampling method are not accurate as theoretical continuous analysis, and it's possible to miss the optimal value. But in a lot of engineering application, continuous analysis is almost impossible to use to solve the problem; the applications of integration is a good and simple example here, and the true optimal value also has no practical meaning if people should pay a lot to get that value.

What we can do is try our best to close the best value we can get. Then people turn to use other methods to solve engineering problems, such as numerical analysis, FEA, sampling methods, etc.

The compromise Decision Support Problem of this case is solved by sampling methods. We assume each sample of the population (the set of individuals, items, or data from which a statistical sample is taken.) has an equal and known chance of being selected. Each sample that is the combination from different design variables represents one possible product in the theoretical product platform. If the entire population will be

sufficiently large, then we will get a more efficient result and have a high probability to get the optimal value. There are some commercial optimization software packages to solve this problem, such as OptdesX, but even for this commercial optimization software, it also requires the user to write and compile a program that contains (or calls) the user's engineering model, which may be written in either C or Fortran.

The algorithm shown in Figure 4.9 can realize the SPPBRD model. It was coded in C++ language. Computations were carried out using an Intel Pentium 4, 1.6 GHz, and 512MB RAM computer. The program contains about 1300 lines codes.

The results of solving the algorithm given in figure 4.9 are tabulated in Table 4.1 and Table 4.2, in which the identified mean and standard deviation of the design variables and resulting performance variations are provided, respectively. The different degree to which the commonality goal is satisfied by the different design variables provides an indication about the system variables that should compose the product platform. The decision on how much variation is negligible is problem specific; however, the value of the standard deviation as a percentage of the mean value can provide a good indication.

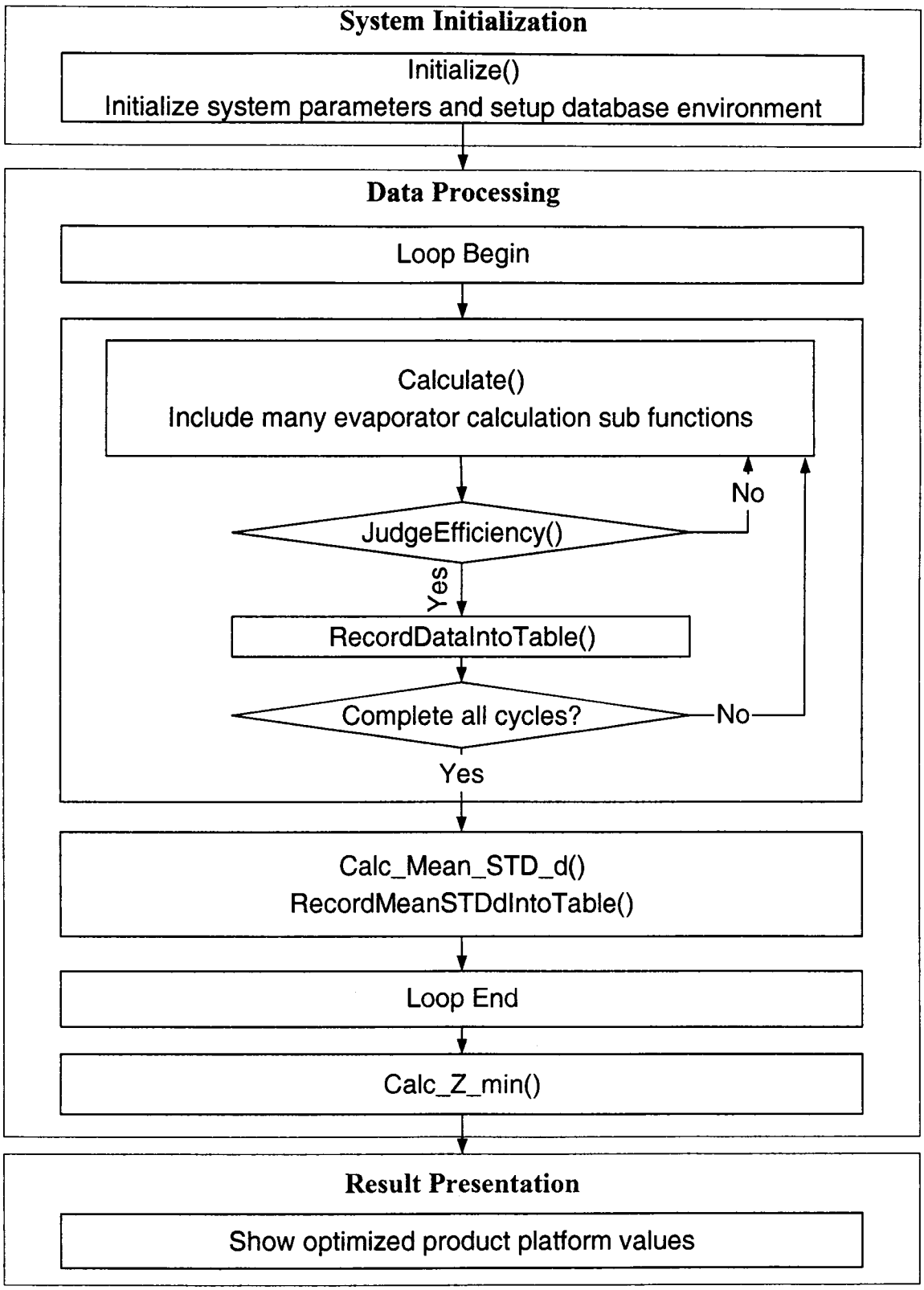

Figure 4.9. Flowchart of solution program

For this problem, standard deviations that are less than 10% of the mean value are considered to be small enough for the corresponding design variable to be considered a common platform parameter.

Based on the results in Table 4.1, the product platform is comprised of: tube spacing S_1, fin spacing b, the air velocity through the narrowest cross-section V_{ac}, air outlet temperature t_2, air volumetric flow rate $\dot{V}$ and the total length of tube L, the width B and thickness of the evaporator Y. Under the assumption of the raw material and specification of the fin and tube, the flow rate V_{ac}, air outlet temperature t_2 and air volumetric flow rate $\dot{V}$ are allowed to vary from evaporator to evaporator within the platform and tube spacing and fin spacing can keep constant, which also fit the real situation that it is hard for punch machine to change tools for different evaporators. From Table 2, the fin efficiency requirement satisfies the mean with a deviation of 2.6% but with a violation of (0.8) 5% from the target. Total heat transfer efficient is also close to the mean value with deviation of 12.9%. However, there is a little bit of difference between the requirement and calculation results of mean and standard deviation of the volumetric flow rate. It is just because considering the more accurate results, I set 15 intervals in the range of air volumetric flow rate in the code that are not those ten given data in the requirements, so that the mean and standard deviation definitely will show a difference between the results and requirements. In stage 2, I will take those ten data in requirements as input parameters to instantiate the evaporator platform. From Table 4.2, the ratio between the standard deviation and mean of the cost is high because this ratio of tube total length, evaporator's width and thickness is also high.

Name	Mean	Standard deviation	Ratio
S_1(m)	0.045	0.0000000225	0
b(m)	0.005	0.001	0.2%
t_2(°C)	11.5	2.31	20.9%
V_{ac}(m/s)	5	0.8	16%
$\dot{V}(m^3/h)$	3000	526	17.5%
L(m)	37.78	26	41.94%
B(m)	0.31	0.14	45.2%
Y(m)	0.25	0.05	20%

Table.4.1 Identified mean and standard deviation of design variables

Name	Mean	Standard deviation
Cost ($)	108.61	67.92
η_f	0.76	0.02
Q_0(kW)	10.4	5.1
$\dot{W}$(g/s)	43.5	25.5
λ(W/m·° C)	27.0	3.5

Table 4.2 Performance parameters for the evaporator platform

In the real application, the bounds and constraints such as the volumetric flow rate and air outlet temperature, market price of raw materials may fluctuate. The program provides an interface to customer shown in Figure 4.10. The user just needs to input the design parameters bounds and other data then click the "Calculate" button. Once the application has completed processing the model, it will report the calculate result in the interface, which will help the designer to choose the product type and make decision.

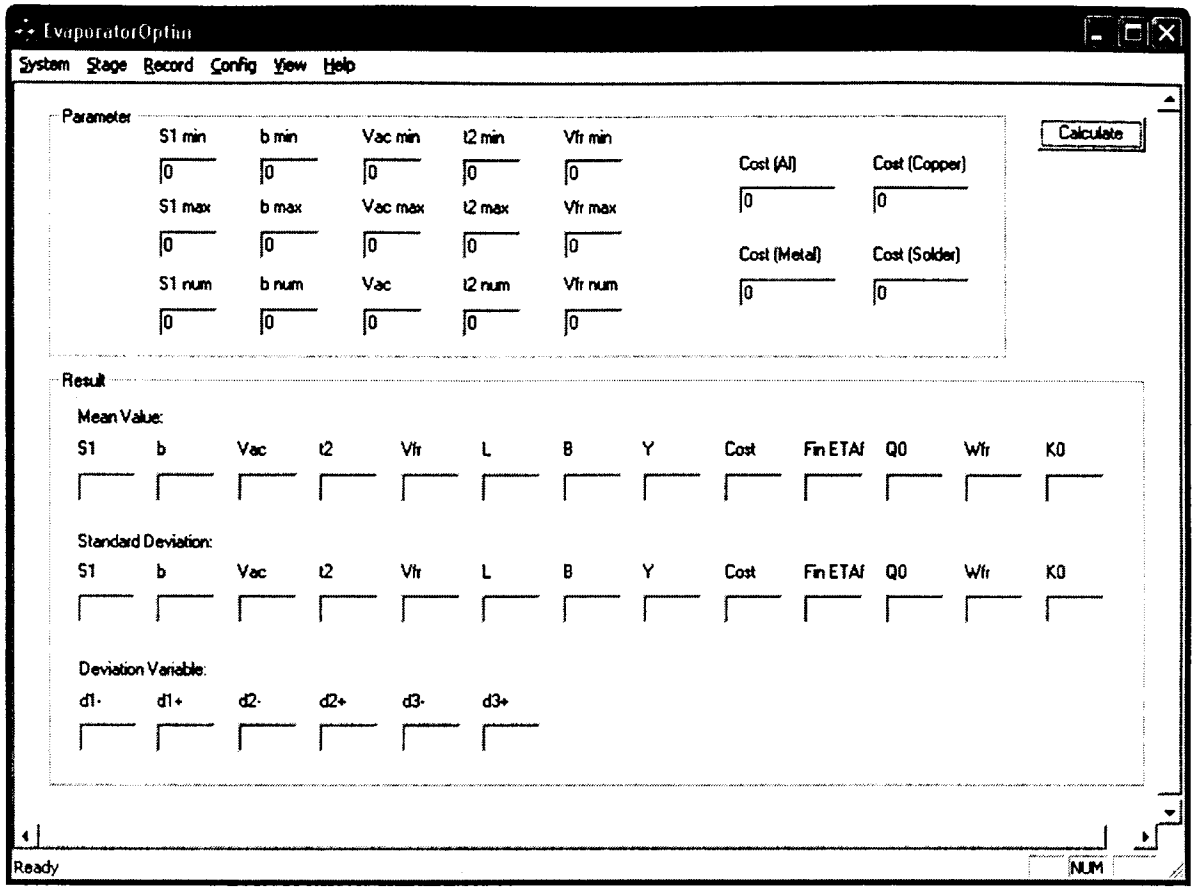

Figure 4.10. The Report menu commands

4.3.4 Stage 2: Create Finned Tube Evaporator Platform

The second stage of the SPPBRD is to instantiate the individual evaporators of the product family using the specifications for the common parameters that describe the product platform. The compromise DSP formulation for designing the individual evaporators is given in Figure 4.11. While the common platform parameters determined from the stage 1 are fixed as constant parameters, the to-be-identified variables are the four remaining non-platform variables and other performance parameters: the tube flow rate through the narrowest cross-section V_{ac}, air outlet temperature t_2, air volumetric flow rate $\dot{V}$ and the total length of tube, the width and thickness of the evaporator. The

constraints and goals are related to the design requirements originally stated at the beginning of section 4.3.3 step 1.Note that the commonality goal is not utilized during the second stage of the SPPBRD since the product platform has already been determined. The algorithm is similar with that in stage 1. The product platform thus developed, represented by the values of all design variables and resulting performance, is listed in Table 4.3.

Given:

Evaporator's mathematical models for design variables (see Section 4.2)

Find: (Solved once for each of j=1, … ,10)

The design variables:

Tube spacing, $S_{1,j}$

Fin pith, b_j

Air velocity of the narrowest cross-section, $V_{ac,j}$

The total valid length of the copper tube, L_j

Width and thickness of evaporator, B_j, Y_j

Air outlet temperature, $t_{2,j}$

Satisfy:

- **System constraints:**

Air volumetric flow rate is given by the set:

$\{1000,1500,1900,2400,3000,3400,3800,4200,4600,5000\}\ m^3/h$

Air inlet and outlet relative humidity: $\varphi_1 = 60 \pm 10\%$, $\varphi_2 = 95\%$

*Inequality constrai*nt on fin efficiency: $0.7 \leq \eta_f \leq 0.8$

Air outlet temperature: $8 \leq t_2 \leq 17\,(^{\circ}C)$

- **System goals**:

$$Cost/IC + d_1^- - d_1^+ = 1$$

$$\eta_f / 0.8 + d_2^- - d_2^+ = 1$$

- **Bounds:**

$$d_1^- \cdot d_1^+ = 0; d_1^-, d_1^+ \geq 0; d_2^- \cdot d_2^+ = 0; d_2^-, d_2^+ \geq 0;$$

$$3mm \leq b \leq 6mm$$

$$30mm \leq S_1 \leq 60mm$$

$$3\ m/s \leq V_{ac} \leq 6\ m/s$$

Minimize

$$Z = \sum_{i=1}^{2} w_i (d_i^- + d_i^+); \quad \sum_{i=1}^{2} w_i = 1; w_i \geq 0$$

Figure 4.11. Compromise DSP to instantiate 10 evaporators from the platform

Product	$\dot{V}$	S_1	b	Vac	t_2	L	B_WIDTH	Y	Cost	η_f	Q_0	$\dot{W}$	λ
1	5000	0.045	0.005	6	11.5	81.11	0.487	0.303	162.7	0.797	17.31	73.25	39.22
2	4600	0.045	0.005	5	11.5	57.93	0.472	0.249	159.3	0.763	15.93	67.38	28.45
3	4200	0.045	0.005	6	11.5	68.13	0.409	0.303	138.9	0.797	14.54	61.50	38.63
4	3800	0.045	0.005	5	11.5	47.86	0.390	0.249	134.0	0.763	13.16	55.50	27.95
5	3400	0.045	0.005	5	11.5	42.82	0.349	0.249	121.3	0.763	11.77	49.75	27.66
6	3000	0.045	0.005	5	11.5	37.78	0.308	0.249	108.6	0.763	10.39	43.63	27.33
7	2400	0.045	0.005	5	11.5	30.23	0.246	0.249	89.6	0.763	8.31	34.88	26.74
8	1900	0.045	0.005	6	11.5	30.82	0.185	0.303	70.6	0.797	6.58	27.38	35.93
9	1500	0.045	0.005	6	11.5	24.33	0.146	0.303	58.7	0.797	5.19	21.63	35.12
10	1000	0.045	0.005	5	11.5	12.59	0.103	0.249	45.2	0.763	3.46	14.25	24.44

Table 4.3 Evaporator product platform instantiated by the SPPBRD

4.4 Analysis and Summary

The algorithm model of the compromise decision support problem considers all possible combinations of design variables. If the design variables $S_1, b, \dot{V}, v_{ac}, t_2$ have m, n, p, q, r intervals in the bounds respectively, then there will be $m \times n \times p \times q \times r$ combinations of all design variables and each combination represents one possible product in the product family and one iteration of the algorithm. Because the air conditioner in this case is not a precision instrument and there is not a very high requirement for precision, we should consider the practical application and manufacturer's existing equipment limitation to give an acceptable value for m, n, p, q, r and we don't need to put a very large number for them, such as the fin spacing and tube spacing, which are limited by the punch machine. This point also gives convincing support for the use of sampling methods to solve the compromise DSP model. In the data processing, those combinations that can't satisfy the constraints (such as fin efficiency) and bounds are removed from the Access database. Every combination left in the database represents one possible product that can satisfy the constraints and bounds and may not be the optimal selection according to the three objectives, but they are still useful for the product selection and analysis when the customer needs to design a new individual evaporator.

According to the result of stage 1, the tube spacing and fin spacing can be seen as constants. So, the air velocity through the narrowest cross-section V_{ac}, air outlet temperature t_2, and air volumetric flow rate $\dot{V}$ are allowed to vary in the range.

Figure 4.12 shows the relationship of evaporator loads and raw materials cost on different air outlet temperatures. It narrates that at the same loads, higher air outlet temperature will save more raw materials, and so our design principle is to set as high outlet temperature as possible to get the same results. It also tells you that with low air outlet temperature, the loads will lie on cost more than those with high air outlet temperature.

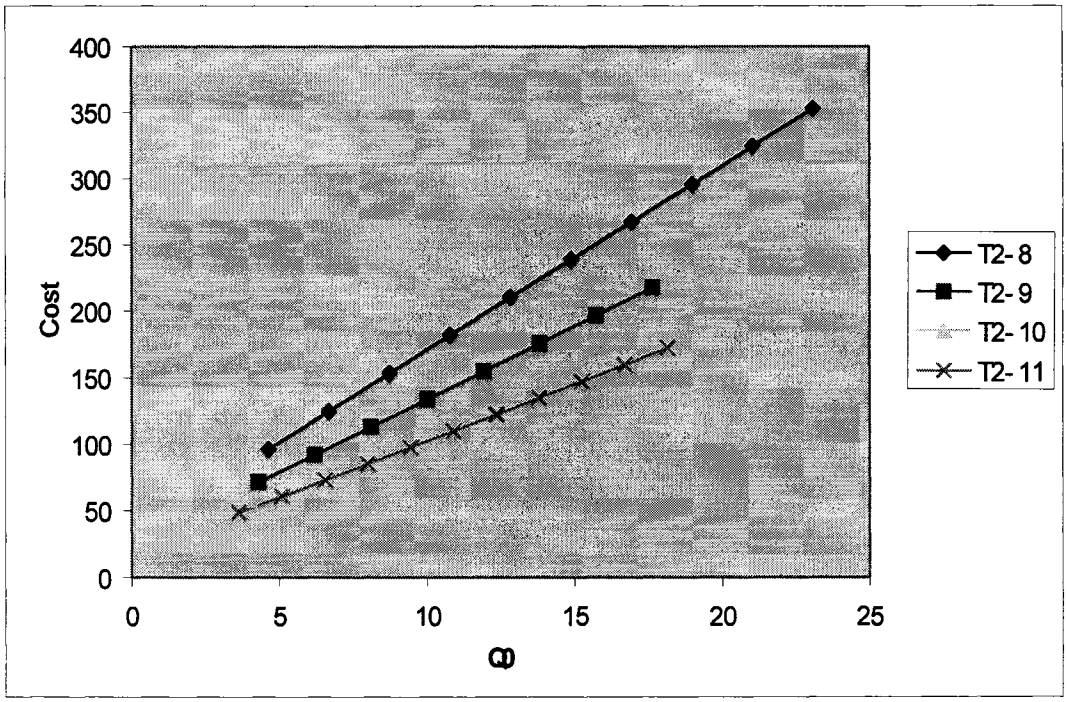

Figure 4.12 Evaporator loads and Cost based on air outlet temperature

The figure 4.13 shows the relationship among the total heat transfer coefficient, fin efficiency and volumetric flow rate with the same air inlet and outlet temperature, and tube and fin specification. When the fin efficiency increases to some extent, the overall heat transfer coefficient will increase very slowly, but different volumetric flow rate can affect the heat transfer coefficient obviously. Actually, the increase of overall heat

transfer coefficient just has a very small effect on the fin efficiency. Both the heat transfer coefficient and fin efficiency depend on material, fin and tube structure and air velocity more than other factors.

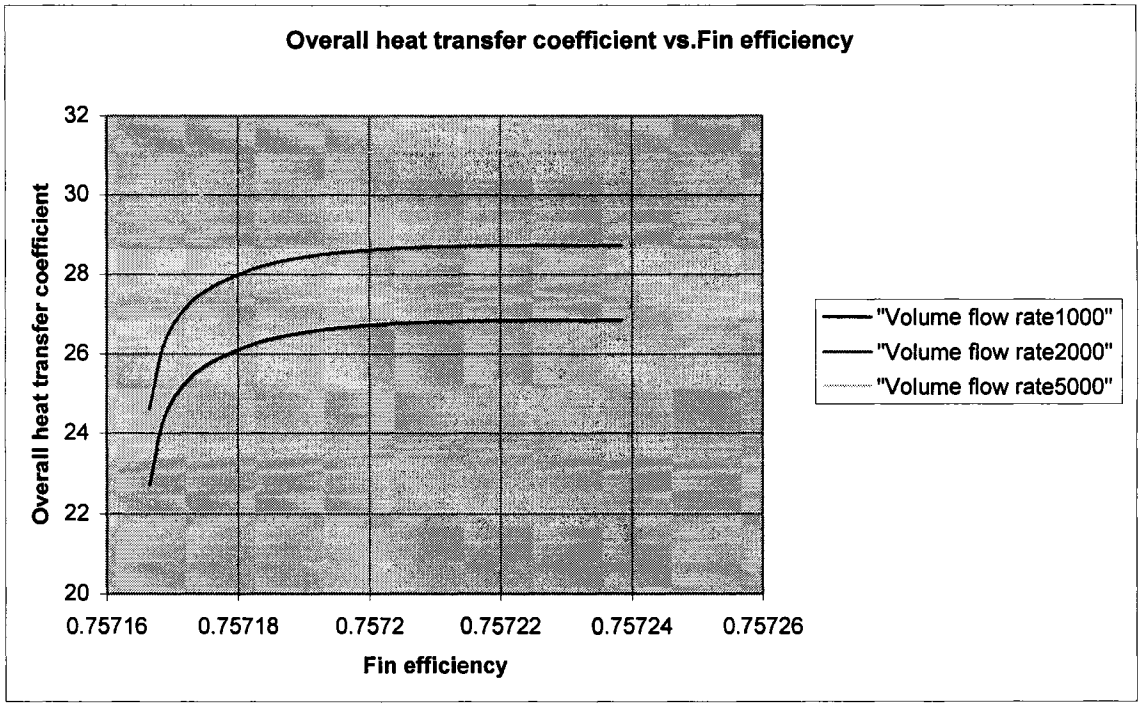

Figure 4.13 Relationship of fin efficiency and overall heat transfer coefficient

Figure 4.14 shows the relationship among the air outlet temperature, fin efficiency and air volumetric flow rate. From this figure we can know the volumetric flow rate has no obviously effects on fin efficiency, the air output temperature does have an effect but not so much because the Y axis increases very slowly with the X axis.

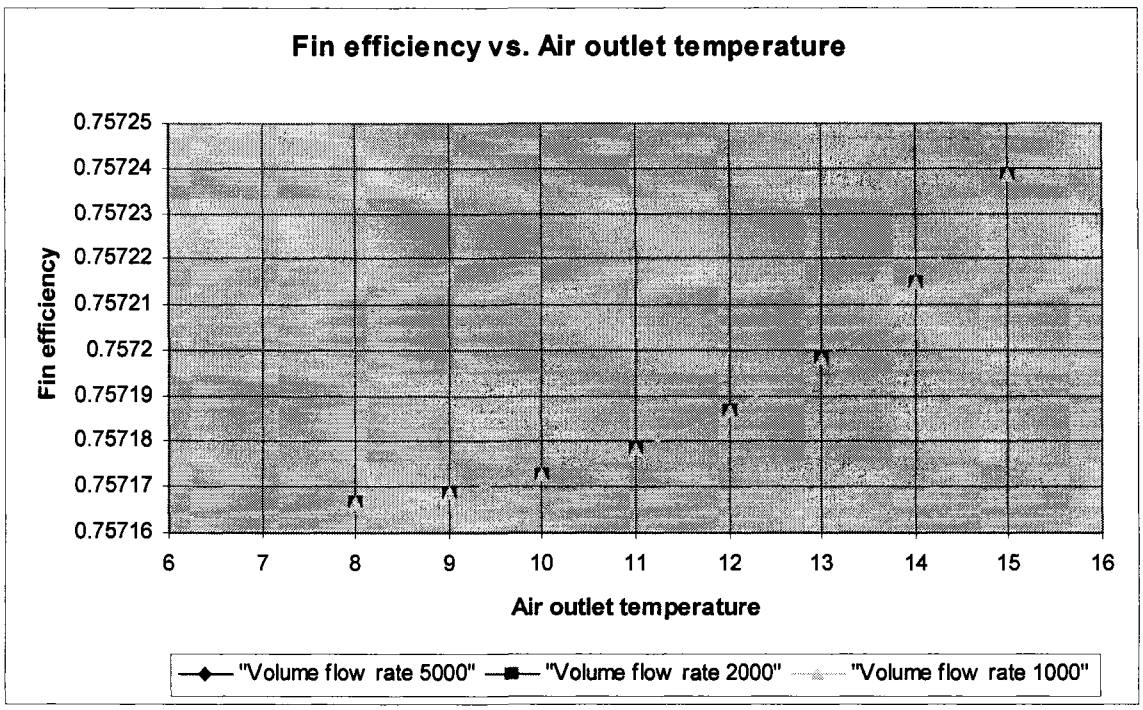

Figure 4.14 Relationship of Air outlet temperature and fin efficiency

This evaporator platform design is not a real case, but it can be used in practical design and production except for the change of some parameters according to real requirements, such as the market price of the raw material or customer's requirement for certain design variables or adding some constraints. This kind of evaporator is not a terminate product for customers; it works with other components (such as fan, motor, panel, and so on) of air conditioner. The visual application interface provides a very convenient and flexible tool for the designer and also helps decision makers to solve problems and get solutions with just a matter of clicks. As some parameters are set in the code and can't change through the visual interface, if the users want to change these kinds of parameters, the code needs some small changes for the detailed requirements.

CHAPTER 5 CONCLUSION

5.1 Conclusion

This thesis provides a systematic method to design product and a more flexible methodology to design scalable product platform. The key issue in the thesis that has been addressed in developing a new design method is to integrate Compromise Decision Support Problem, Robust Design, and Two-stage Approaches into a decision model. From the literature review and the methodology it is concluded that the integration of Compromise Decision Support Problem, Robust design is a very important and useful tool to solve the multiple objective problems and give an optimal design for the platform at the same time. This method works through the conceptual product design process, the detailed product design to the product family development. At the conceptual design stage, the biggest challenge for designers is not only to generate and evaluate the alterative ideas that may form the objectives of the whole product platform or work as the known requirements of the whole design, but also list overall requirements for the design and try to identify the market segmentation for the product platform and then classify all design parameters.

The SPPBRD uses a two-stage approach to design and develop a product platform and corresponding family of products. In the first stage the SPPBRD uses the compromise DSP to make tradeoffs between commonality and individual product performance. In this stage, the infusion of Taguchi's Robust design into Compromise Decision Support Problem is the key point. The decision model facilitates the design of a common product platform that can be scaled to realize a product family. The meaning of the common product platform is significant because it will simplify the future design of

the product platform and can reduce some design parameters. Through the identification of appropriate non-platform variables (scaling factors) during the product platform design process, the individual targets for derivative product can be aggregated into a mean and variance around which the product platform can be simultaneously designed either by having separate goals to "bringing the mean on target" and "minimizing the variation" to measure the capability of a family of designs to satisfy a ranged set of design requirements. In stage 2, the common product platform is used as the core to develop the individual products of the platform, i.e., to determine the values of non-platform variables. Then, the designer can design individual products in the platform very easily according to customers' requirements; and at the same time all possible optimal configurations for this individual product has been considered.

Application of the method is demonstrated by means of on example: a platform of 10 finned tube evaporators that have a dehumidification effect. This example integrates nicely within the framework of the SPPBRD, and the attached code for this evaporator platform can be used to design and produce evaporator immediately without any modification if in the real application the requirements are same as those in this thesis. Even though the methodology is only demonstrated for this example, it is asserted that the method is generally applicable to other examples in this class of problems: parametrically scalable product platforms whose performance can be mathematically modeled or simulated.

5.2 Contribution

I identify the contributions of this research work as following:

- Understanding the product platform concept and providing a decision model that combines the compromise decision support problem, two-stage approaches, robust design, and multiple objectives to design scalable product platform
- The SPPBRD decision model can be widely used in different fields: parametrically scalable product platforms whose performance can be mathematically modeled or simulated
- The author is the first user of the compromise Decision Support Problem model to design product platform for evaporators with dehumidification effects
- The software package for evaporator platform can be used in practical design and production to reduce the lead-time, improve the designer' ability to develop new product and save design, manufacturing and raw material cost
- Using P-diagram to classify the design parameters for evaporators design that can simplify the designer's logic and get the main point of design quickly
- Infusing robust design concept into evaporator design and making the evaporator design lean and robust

5.3 Recommendations for future research

The present work may be further extended in a number of ways outlined below.

The present application of the model is limited to scalable product platform or scalable product components. In another word, variations of product functionality are not considered. Therefore, one possibility that can be investigated is applying the proposed theory and method independently to functional modules (modular products). Another possibility is that the investigation of a method for platform scaling is suitable for products with complex integral architecture. Further research is required to extend the

applicability of the method for designing large systems like automobile or aircraft that involve many sub-systems.

The decision model just considers customers' requirement about the change of design parameters and doesn't include the uncertain distributions of demand, market price change for raw material, order cost, equipment cost, labor cost and plant cost, in practical production. All these can affect the manufacturer's decision. In future work, these can be a good point to improve the model. But the evaporator platform software package does consider this point.

The decision model itself has many places that need improvement or more study on that. As well, in the conceptual design process, there are many uncertain factors that are hard to control, such as classifying the design parameters, assessing weights to different system goals, developing more tools, and doing more research on that to reduce the uncertain factors are very important for the product family and platform design.

Additionally, assessing weights to commonality goal and different system goals is an important issue that will affect the results of the decision model. Even there are a lot of theoretical research on how to assess weight that has been addressed on section 3.2.1 step 5, however, there are no related research and experiment data of the attributes' weights for the finned tube evaporator design. Future work may put more effort on preparing and collecting data for weights of different goals for finned tube evaporator.

Applying SPPBRD to other real product platform design and manufacturing process are promising and can be expected. The result of using this new approach is worth for researcher to compare them with results using other models. From comparison, the advantages, application fields of this model will be classified and identified.

The SPPBRD is not an end in itself; rather, it provides a stepping-stone for future research work in this nascent field of engineering design. For it is only at the end of this thesis that the problems and difficulties associated with product family and product platform design are truly understood and appreciated. Now that we understand them, either for the first time or in greater depth, new paths can be explored or new methods can be developed which continue to advance the state-of-the-art in product family and product platform design. It is the hope of the author that the SPPBRD enjoys the same success as the other product platform model, providing a foundation on which future research can be established in the same way that this work has built upon the work before it.

REFERENCE

1. Arthur, P., 1988, "Heat Exchanger", A Wiley-Interscience Publication, John Wiley & Sons, Inc. New York, ISBN 0-471-62868-9
2. ASHRAE Systems And Equipment Handbook, 2000, HVAC System Analysis And Selection, Chapter 43
3. Balaji C., Stone, R., Mcadams, D., 2004,"Developing Design Templates for Product Platform Focused Design", Journal of Engineering Design, Vol. 15, No. 3, pp 209 -228
4. Caffrey, R., Mitchell, G., Wahl, Z., Zenick, R., 2002, "Product Platform Concepts Applied To Small Satellites: A Radio Concept By Aeroastro Inc.", 16th Annual Conference On Small Satellites, Logan, USA
5. Chen, W., Wiecek, M., Zhang, J., 1998, "Quality Utility – A Compromise Programming Approach To Robust Design", Modified Manuscript to JMD, Department of Mechanical Engineering, University of Illinois, Chicago, USA
6. Fellini, R., Papalambros, P., Weber, T., 2000, "Application of Product Platform Design Process to Automotive Powertrains", University of Michigan, Ann Arbour, USA
7. Fellini, R., Sasena, M., Papalambros, P., 2002,"Optimal Design Decisions in Product Portfolio Valuation", 2002, Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Canada
8. Fujita, K., Yoshida, H., 2001,"Product Variety Optimization: Simultaneous Optimization of Module Combination and Module Attributes", 2001, Design Engineering Technical Conferences And Computers And Information In Engineering Conference, Pittsburgh, USA
9. Gao, J., Aziz, H., Maropoulos, P., Cheung, W., 2003, "Application of Product Data Management Technologies for Enterprise Integration" International Journal of. Computer Integrated Manufacturing, Vol. 16, No. 7 - 8, pp 491 - 500

10. Governor, A., 2004,"Prototype Design, Testing, and Analysis", Technical Report, California Energy Commission, Cambridge, MA, USA

11. Gupta, S., Krishnan, V., 1998, "Product Family-Based Assembly Sequence Design Methodology", IIE Transactions, vol.30, No.10, pp.933-945

12. Kakac, S., Liu, H., 2002, "Heat Exchangers Selection, Rating, and Thermal Design", CRC Press, Boca Raton, FL, ISBN: 0-8493-0902-6

13. Koen, P. A., 2004,"The Fuzzy Front End for Incremental, Platform And Breakthrough Products and Services", The PDMA Handbook of New Product Development, Hoboken, NJ, ISBN: 0-471-48524-1

14. Kokkolaras, M., Fellini, R., Kim, H., Michelena, N., Papalambros, P., 2002, "Extension of the Target Cascading Formulation to the Design of Product Families", Struct Multidisc Optimal, vol. 24, pp 293–301

15. Kulkarni, R., 2005, " Infusion of Robustness into the Product Platform Construct Theory Method ", Master Thesis, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, USA

16. Mattson, C., Messac, A., 2004, "Case Studies in Concept Selection Using S-Pareto Frontiers", Design and Optimization Symposium, Department of Mechanical, Aerospace, and Nuclear Engineering, Rensselaer Polytechnic Institute, NY, USA

17. Messac, A., Martinez, M., Simpson, T., 2002, "Introduction of a Product Family Penalty Function Using Physical Programming", Journal of Mechanical Design, Vol.124, No.2, pp 164-172

18. Messac, A., Martinez, M., Simpson, T., 2002,"Effective Product Family Design Using Physical Programming", Engineering Optimization, Vol. 124, No. 3, pp 245-261

19. Mistree, F., Hughes, O., Bras, B., 1993, "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm", Structural Optimization: Status and Promise, MP Kamat, ed., AIAA, Washington, DC, pp 247-286

20. Muller, G., 2004,"Product Family Business Analysis and Definition", Preliminary Draft, Embedded Systems Institute, Eindhoven, Netherlands

21. Nayak, R., Chen, W., Allen, J., Mistree, F., 1999, "A Variation-Based Methodology for Product Family Design," Proceedings of the 2000 ASME Design Automation Conference, Baltimore, USA

22. Nayaka, R., Chen, W., Simson, T., 2001, " A Variation-Based Method for Product Family Design", Engineering Optimization, Vol. 34(1), pp 65–81

23. NX Digital Product Development White Paper, 2003,Business Process Initiative: Design For Six Sigma

24. Park, S., "Six Sigma for Quality and Productivity Promotion", 2003, The Asian Productivity Organization, Tokyo, ISBN: 92-833-1722-X

25. Proctor, J., Albright, P., 1995, "Sizing Air Conditioners: If Bigger is Not Better, What is?" Home Energy Magazine, HE May/June '95, p. 19

26. Sanderson, S., Uzumeri, M., 1995, "Managing Product Families: The Case of The Sony Walkman", working paper, School of Management, Rensselaer Polytechnic Institute, Troy, USA

27. Seepersad, C., 2001, "The Utility-Based Compromise Decision Support Problem with Applications in Product Platform Design", working paper, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, USA

28. Seepersad, C., Cowan F., Chamberlain M., Mistree F., 2002, "Strategic Design: Leveraging and Innovation for a Changing Marketplace," Computer-Based design: Engineering Design Conference, London, UK

29. Seepersad, C., Hernandez, G., Allen, J., 2000, "A Quantitative Approach to Determining Product Platform Extent", ASME Advances in Design Automation Conference, Baltimore, USA

30. Seepersad, C., Mistree, F., Allen, J., 2002, "A Quantitative Approach for Designing Multiple Product Platforms for an Evolving Portfolio of Products", ASME Advances in Design Automation Conferences, Montreal, Canada

31. Simpson, T., 2003, "Product Platform Design and Optimization: Status and Promise", ASME Design Engineering Technical Conferences, Chicago, USA

32. Simpson, T., Chen, W., Allen, J., Mistree, F., 1996, "Conceptual Design of A Family of Products through the Use of the Robust Concept Exploration Method", 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis And Optimization, Bellevue, USA

33. Simpson, T., Chen, W., Allen, J., Mistree, F., 1999, "Use of the Robust Concept Exploration Method to Facilitate the Design of a Family of Products", Simultaneous Engineering: Methodologies and Applications, pp 247-278

34. Simpson, T., "Product Platform Design and Customization: Status and Promise", 2004, Special Issue: Platform Product Development For Mass Customization, Vol.18, No.1

35. Simpson, T., Maier, J., Mistree, F., 1999, "A Product Platform Concept Exploration Method for Product Family Design", ASME Design Theory and Methodology Conference, Las Vegas, USA

36. Simpson, T., D'souza, B., 2002,"Assessing Variable Levels of Platform Commonality within a Product Family Using a Multi-Objective Genetic Algorithm", 9th AIAA/ISSMO Symposium On Multidisciplinary Analysis and Optimization, Pennsylvania, USA

37. Sopadang, A., Cho, B., 1999, "A Framework for Product Family Based Robust Design", Working Paper, Department of Industrial Engineering, Clemson University, USA

38. Sopadang, A, Kim, Y., Cho, B., 2000, "A Method for Assessing Weight of Multiple Quality Attributes", Department of Industrial Engineering, Clemson University, USA

39. Sopadang, A, Teeravaraprug, J, Cho, B., 2002, "Overview of Robust Design within the Framework of Product Family", Department of Industrial Engineering, Clemson University, USA

40. Sopadang, A., Cho, B., 1999, "Attribute Ranking Analysis for Product Family Based Robust Design", Working Paper, Department of Industrial Engineering, Clemson University, USA

41. Sopadang, A., Cho, B., 2000, "Scaling Factor Identification for Product Family Based Robust Design", Working Paper, Department of Industrial Engineering, Clemson University, USA

42. Taborek, J., Hewitt, G., Afgan, N., 1983, "Heat Exchangers Theory And Practice", Hemisphere Publishing Corporation, ISBN0-07-062806-8

43. Tsui, K., 1992, "An Overview of Taguchi Method and Newly Developed Statistical Methods for Robust Design", IIE Transactions, Vol. 24, No. 5, pp 44-57

44. Weck, O., Eun, S., 2003, "Product Family and Platform Portfolio Optimization", ASME Design Engineering Technical Conferences, Cambridge, USA

45. Williams, C., 2003, " Platform Design for Customizable Products and Processes with Non-Uniform Demand ", Mater Thesis, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, USA

46. Williams, C., Allen, J., Rosen, D., Mistree, F., 2004, "Designing Platforms for Customizable Products in Markets with Non-Uniform Demand", Design Theory and Methodology Conference, Salt Lake City, USA

47. William, R., Apblett, J., 1981, "Second Symposium on Shell and Tube Heat Exchangers", American Society For Metals, Houston, Texas, ISBN:0-87170-145-6

48. Wu, Y., Han B., 2004, "Refrigeration Principle and Equipments", Xi'an Transportation University Puslication, Xi'an, China, ISBN: 7-5605-0059-5/Tb.3

49. Xue, D., 1999, "Air-Conditioning", Tsinghua University Publication, Beijing, China, ISBN: 70302-00805-1/Tu-65

50. www.Prtm.Com/Services/One.Asp? Service_Id=6481, Retrieved on Oct.20, 2005

51. www.doi.eng.cmu.ac.th/~apichat/family.html. Retrieved on Sept.2, 2004

52. www.doi.eng.cmu.ac.th/~apichat/intro.html. Retrieved on Dec.10, 2004

APPENDIX

C++ CODE

```
// DataStruct.h :  Define struct used in Calculation
//
////////////////////////////////////////////////////////////////////////////

struct EVP_DATA
{
        float   S1;                     //Tube spacing
        float   b;                      //Fin spacing
        float   Vac;                    //Air Velocity
        float   t2;                     //Air outlet Temperature
        float   Vfr;                    //Volume Flow Rate
        float   L;                      //Tube total Length
        float   B;                      //Width of Evaporator
        float   Y;                      //Thickness of Evaporator
        float   Cost;                   //Cost of Evaporator
        float   ETAf;                   //Fin Efficiency
        float   Q0;                     //Evaporator loads
        float   Wfr;                    //Moisture Removal Flow Rate
        float   K0;                     //Overall heat transfer coefficient
        float   IC;                     //Ideal Cost
        int  DATA_VALID;        //1 indicates valid, 0 indicates invalid
};


struct INPUT
{
        float   S1_min;         //Min Tube Spacing
        float   S1_max;         //Max Tube Spacing
        int     S1_num;         //Number of S1
        float   S1_step;        //Step increase size of S1
        float   b_min;          //Min Fin Pitch
        float   b_max;          //Max Fin Pitch
        int     b_num;          //Number of b
        float   b_step;         //Step increase size of b
        float   Vac_min;        //Min Velocity of smallest cross section
        float   Vac_max;        //Max Velocity of smallest cross section
        int     Vac_num;        //Number of Vac
        float   Vac_step;       //Step increase size of Vac
        float   t2_min;         //Min Air Outlet Temperature
```

```
        float   t2_max;         //Max Air Outlet Temperature
        int     t2_num;         //Number of t2
        float   t2_step;        //Step increase size of t2
        float   Vfr_min;        //Min Volume Flow Rate
        float   Vfr_max;        //Max Volume Flow Rate
        int     Vfr_num;        //Number of Vfr
        float   Vfr_step;       //Step increase size of Vfr
        float   C_Aluminum;     //Cost of Aluminum fin
        float   C_Copper;       //Cost of Copper tube
        float   C_Metal;        //Cost of Sheet metal
        float   C_Solder;       //Cost of Solder
};

struct OUTPUT
{
        float   Mu_S1;
        float   Mu_b;
        float   Mu_Vac;
        float   Mu_t2;
        float   Mu_Vfr;         //Mean Value of Volume Flow Rate
        float   Mu_L;
        float   Mu_B;
        float   Mu_Y;
        float   Mu_Cost;
        float   Mu_ETAf;
        float   Mu_Q0;
        float   Mu_Wfr;
        float   Mu_K0;

        float   Sigma_S1;
        float   Sigma_b;
        float   Sigma_Vac;
        float   Sigma_t2;
        float   Sigma_Vfr;      //Standard deviation of Volume Flow Rate
        float   Sigma_L;
        float   Sigma_B;
        float   Sigma_Y;
        float   Sigma_Cost;
        float   Sigma_ETAf;
        float   Sigma_Q0;
        float   Sigma_Wfr;
        float   Sigma_K0;

        float   d1_NEG;         //d1-
        float   d1_POS;         //d1+
        float   d2_NEG;
```

```
        float   d2_POS;
        float   d3_NEG;
        float   d3_POS;
        float   Z;
};



// EvpOptimView.cpp: implementation of the CEvp class.
//
//////////////////////////////////////////////////////////////////////////

struct INPUT in;

void CEvpOptimView::OnCalculate()
{
        // TODO: Add your control notification handler code here
        UpdateData(TRUE);
        in.S1_min       = m_S1_min;
        in.S1_max       = m_S1_max;
        in.S1_num       = m_S1_num;
        in.b_min        = m_b_min;
        in.b_max        = m_b_max;
        in.b_num        = m_b_num;
        in.Vac_min      = m_Vac_min;
        in.Vac_max      = m_Vac_max;
        in.Vac_num      = m_Vac_num;
        in.t2_min       = m_t2_min;
        in.t2_max       = m_t2_max;
        in.t2_num       = m_t2_num;
        in.Vfr_min      = m_Vfr_min;
        in.Vfr_max      = m_Vfr_max;
        in.Vfr_num      = m_Vfr_num;
        in.C_Aluminum= m_Cost_Al;
        in.C_Copper     = m_Cost_Cu;
        in.C_Metal      = m_Cost_Metal;
        in.C_Solder     = m_Cost_Solder;

        int rtn;
        struct OUTPUT result;

        result.Mu_S1            = 0.0f;
        result.Mu_b             = 0.0f;
        result.Mu_Vac           = 0.0f;
        result.Mu_t2            = 0.0f;
        result.Mu_Vfr           = 0.0f;
```

```
result.Mu_L          = 0.0f;
result.Mu_B          = 0.0f;
result.Mu_Y          = 0.0f;
result.Mu_Cost       = 0.0f;
result.Mu_ETAf       = 0.0f;
result.Mu_Q0         = 0.0f;
result.Mu_Wfr        = 0.0f;
result.Mu_K0         = 0.0f;
result.Sigma_S1      = 0.0f;
result.Sigma_b       = 0.0f;
result.Sigma_Vac     = 0.0f;
result.Sigma_t2      = 0.0f;
result.Sigma_Vfr     = 0.0f;
result.Sigma_L       = 0.0f;
result.Sigma_B       = 0.0f;
result.Sigma_Y       = 0.0f;
result.Sigma_Cost    = 0.0f;
result.Sigma_ETAf    = 0.0f;
result.Sigma_Q0      = 0.0f;
result.Sigma_Wfr     = 0.0f;
result.Sigma_K0      = 0.0f;
result.d1_NEG        = 0.0f;
result.d1_POS        = 0.0f;
result.d2_NEG        = 0.0f;
result.d2_POS        = 0.0f;
result.d3_NEG        = 0.0f;
result.d3_POS        = 0.0f;

CEvp evp;
rtn = evp.Initialize(&in);
rtn = evp.Process(&result);

m_Mu_S1.Format("%.4f", result.Mu_S1);
m_Mu_b.Format("%.3f", result.Mu_b);
m_Mu_Vac.Format("%.1f", result.Mu_Vac);
m_Mu_t2.Format("%.1f", result.Mu_t2);
m_Mu_Vfr.Format("%.0f", result.Mu_Vfr);
m_Mu_L.Format("%.2f", result.Mu_L);
m_Mu_B.Format("%.2f", result.Mu_B);
m_Mu_Y.Format("%.2f", result.Mu_Y);
m_Mu_Cost.Format("%.2f", result.Mu_Cost);
m_Mu_ETAf.Format("%.2f", result.Mu_ETAf);
m_Mu_Q0.Format("%.1f", result.Mu_Q0);
m_Mu_Wfr.Format("%.1f", result.Mu_Wfr);
m_Mu_K0.Format("%.1f", result.Mu_K0);
m_Sigma_S1.Format("%.4f", result.Sigma_S1);
```

```
        m_Sigma_b.Format("%.3f", result.Sigma_b);
        m_Sigma_Vac.Format("%.1f", result.Sigma_Vac);
        m_Sigma_t2.Format("%.1f", result.Sigma_t2);
        m_Sigma_Vfr.Format("%.0f", result.Sigma_Vfr);
        m_Sigma_L.Format("%.2f", result.Sigma_L);
        m_Sigma_B.Format("%.2f", result.Sigma_B);
        m_Sigma_Y.Format("%.2f", result.Sigma_Y);
        m_Sigma_Cost.Format("%.2f", result.Sigma_Cost);
        m_Sigma_ETAf.Format("%.2f", result.Sigma_ETAf);
        m_Sigma_Q0.Format("%.1f", result.Sigma_Q0);
        m_Sigma_Wfr.Format("%.1f", result.Sigma_Wfr);
        m_Sigma_K0.Format("%.1f", result.Sigma_K0);
        m_d1_NEG.Format("%.2f", result.d1_NEG);
        m_d1_POS.Format("%.2f", result.d1_POS);
        m_d2_NEG.Format("%.2f", result.d2_NEG);
        m_d2_POS.Format("%.2f", result.d2_POS);
        m_d3_NEG.Format("%.2f", result.d3_NEG);
        m_d3_POS.Format("%.2f", result.d3_POS);

        UpdateData(FALSE);
}



// Evp.cpp: implementation of the CEvp class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "EvpOptim.h"
#include "Evp.h"

#include <math.h>

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CEvp::CEvp()
{
```

```
        pSequence = new CSequence(NULL);
        pEvpSet = new CEvpOptimSet(NULL);
        pSet = new CMeanSTDd(NULL);
}

CEvp::~CEvp()
{
        delete pEvpSet;
        delete pSequence;
        delete pSet;
}


struct INPUT input;

int CEvp::Initialize(struct INPUT *pIn)                    //Initialize
{
        input.S1_max = pIn->S1_max;
        input.S1_min = pIn->S1_min;
        input.S1_num = pIn->S1_num;
        input.b_max = pIn->b_max;
        input.b_min = pIn->b_min;
        input.b_num = pIn->b_num;
        input.Vac_max = pIn->Vac_max;
        input.Vac_min = pIn->Vac_min;
        input.Vac_num = pIn->Vac_num;
        input.t2_max = pIn->t2_max;
        input.t2_min = pIn->t2_min;
        input.t2_num = pIn->t2_num;
        input.Vfr_max = pIn->Vfr_max;
        input.Vfr_min = pIn->Vfr_min;
        input.Vfr_num = pIn->Vfr_num;
        input.C_Aluminum = pIn->C_Aluminum;
        input.C_Copper = pIn->C_Copper;
        input.C_Metal = pIn->C_Metal;
        input.C_Solder = pIn->C_Solder;

        if (input.S1_num == 1)
                input.S1_step = 0;
        else
                input.S1_step = (input.S1_max - input.S1_min) / (input.S1_num - 1);
        if (input.b_num == 1)
                input.b_step = 0;
        else
                input.b_step = (input.b_max - input.b_min) / (input.b_num - 1);
        if (input.Vac_num == 1)
```

```
            input.Vac_step = 0;
        else
            input.Vac_step = (input.Vac_max - input.Vac_min) / (input.Vac_num - 1);
        if (input.t2_num == 1)
            input.Vac_step = 0;
        else
            input.t2_step = (input.t2_max - input.t2_min) / (input.t2_num - 1);
        if (input.Vfr_num == 1)
            input.Vfr_step = 0;
        else
            input.Vfr_step = (input.Vfr_max - input.Vfr_min) / (input.Vfr_num - 1);

        pSequence->Open();
        pSet->Open();
        pEvpSet->m_strFilter.Format("%S", "NUMBER > 0");
        pEvpSet->m_strSort.Format("%s", "NUMBER ASC");
        pEvpSet->Open();
        pEvpSet->m_pDatabase->BeginTrans();
        TRY
        {
            pEvpSet->m_pDatabase->ExecuteSQL("DELETE * FROM EVP
WHERE NUMBER > 0");
            pEvpSet->m_pDatabase->ExecuteSQL("DELETE * FROM Mean_STD_d
WHERE NUMBER > 0");
            pSequence->m_pDatabase->ExecuteSQL("UPDATE SEQUENCE_D
SET SEQUENCE_NUM = 0 WHERE SEQUENCE_D = \'EVP\'");
            pSequence->m_pDatabase->ExecuteSQL("UPDATE SEQUENCE_D
SET SEQUENCE_NUM = 0 WHERE SEQUENCE_D = \'Mean_STD_d\'");
            pEvpSet->m_pDatabase->CommitTrans();
        }
        CATCH (CDBException, e)
        {
            pEvpSet->m_pDatabase->Rollback();
        }
        END_CATCH
        pSequence->Close();
        pEvpSet->Close();
        pSet->Close();

        return 1;
}


int CEvp::JudgeFinEfficiency(float ETAf)
{
        if (ETAf > 0.80 || ETAf < 0.60) return 0;
```

```
	return 1;
}


int CEvp::Compute_Property(float t, float phi, float *pt_p_qb, float *pt_d, float *pt_h,
float *pt_t_l)
{
	float T, p_qb, d, h, t_l;
	T = 273 + t;
	p_qb = exp(-5800.2206/T + 1.3914993 + (-0.04860239)*T + (0.41764768e-
4)*pow(T, 2) + (-0.14452093e-7)*pow(T, 3) + 6.5459673*log(T));
	d = 0.622 * ( (phi*p_qb)/(101325 - phi*p_qb) );
	h = 1.01*t + 0.001*d*(2501+1.84*t);
	t_l = 8.22 + 1.24*log(phi*p_qb/1000) + 1.9*pow(log(phi*p_qb/1000), 2);

	*pt_p_qb = p_qb;
	*pt_d = d;
	*pt_h = h;
	*pt_t_l = t_l;

	return 1;
}


int CEvp::Calculate(struct EVP_DATA *pEvp)
{
	float S1, b, Vac, t2, Vfr, B, ETAf, Q0, Wfr, L, Y, K0, Cost;
	S1	= pEvp->S1;
	b	= pEvp->b;
	Vac	= pEvp->Vac;
	t2	= pEvp->t2;
	Vfr	= pEvp->Vfr;

	float M = 20;		//Number of tubes for each row;

	//Calculate B		B = f(S1, Vac, b, Vfr)
	float d0, delta;
	d0 = 0.015;
	delta = 0.0003;
	float A_l, A_y;
	A_l = Vfr / (3600*Vac);
	A_y = (A_l * S1 * b) / ( (S1 - d0) * (b - delta) );
	B = A_y / (S1 * M);

	//Calculate ETAf	Etaf = f(S1, Vac, b, t2, Vfr)
	//Compute Alpha_0f
```

```
        float Alpha_0f;
        float lambda = 0.0252, nu = 0.153e-4;
        float h;
        h = 0.5 * (S1 - d0);
        Alpha_0f = 0.205 * (lambda/b) * pow(((Vac * b)/nu), 0.65) * pow((d0/b), -0.4) *
pow((h/b), -0.14);
        //Compute SHR
        float SHR, Cp, t1, phi1, phi2, p_qb1, p_qb2, d1, d2, h1, h2, t_l1, t_l2;
        t1 = 22;
        phi1 = 0.7;
        phi2 = 0.95;
        Compute_Property(t1,  phi1, &p_qb1, &d1, &h1, &t_l1);
        Compute_Property(t2,  phi2, &p_qb2, &d2, &h2, &t_l2);
        Cp = 1.0049 + 1.8842*(d2/1000);
        SHR = Cp * (t2 - t1) / (h2 - h1);
        //Compute ETAf
        float m, lambda_Al, R0, Zeta;
        lambda_Al = 203.5;
        m = pow( ( 2*Alpha_0f/(SHR*lambda_Al*delta) ), 0.5);
        R0 = 0.5 * d0;
        Zeta = (S1/d0 - 1) * (1 + 0.35*log(1.063*S1/d0));
        ETAf = ( tanh(m*R0*Zeta) ) / (m*R0*Zeta);

        //Calculate Q0          Q0 = f(t2, Vfr)
        float v_a, m_a;
        v_a = (287.09 * (273+t1)) / (101325 - phi1*p_qb1);
        m_a = Vfr/v_a;
        Q0 = m_a * (h1 - h2) / 3600;  //Q0 in unit of KW

        //Calculate Wfr                 Wfr = f(t2, Vfr)
        Wfr = (d1 - d2) * m_a * 3600 / 1000;


        //Calculate L           L = f(S1, Vac, b, t2, Vfr)
        //Compute Delta_t0
        float Delta_t0;
        Delta_t0 = (t1-t2) / log( (t1 - (t_l2-1)) / (t2 - (t_l2-1)) );
        //Compute ETA0
        float ETA0, Ar, Ai, Af;
        Ar = 3.14159 * d0 * (1 - delta/b);
        Ai = 3.14159 * (d0 - 0.002) * (1 - delta/b);
        Af = 2 * (S1*S1*1.732/2 - (3.14159/4)*d0*d0) * (1/b);
        ETA0 = (Ar + ETAf*Af) / (Ar + Af);
        //Compute F0f
        float F0f;
        F0f = (Q0*1000*SHR) / (Alpha_0f*Delta_t0*ETA0);
```

```
        //Compute L
        L = F0f / (Ar + Af);
        //Compute N             //Number of tubes
        int N;
        N = int((L/B) * 1.1);
        //Revise F0f
        F0f = N * B * (Ar + Af);
        //Calculate Y   //Thickness of the Evaporator
        Y = (int(N / M)) * S1 * 0.866;
        //Compute Fr, Fi, Ff
        float Fr, Fi, Ff;
        Fr = N * B * Ar;
        Fi = N * B * Ai;
        Ff = N * B * Af;

        //Calculate K0          K0 = f(S1, Vac, b, t2, Vfr, L, B)
        //Compute Alpha_i
        float Alpha_i, v_m, q_i;
        v_m = Q0 / (121.7*M * 3.14159*pow((d0-0.002), 2)/4);
        q_i = Q0*1000 / Fi;
        Alpha_i = 57.8 * 0.0199 * pow(v_m, 0.2) * pow(q_i, 0.6) / pow((d0-0.002), 0.2);
        //Compute K0
        K0 = 1 / ( ( (1/Alpha_i+delta/387) * (F0f/Fi) ) + ( (SHR/Alpha_0f) *
(F0f/(Fr+ETAf*Ff)) ) );

        //Calculate Cost        Cost = f(Vac, S1, b, t2, Vfr, L, B)
        //Compute C_Material
        float C_Material, C_fin, C_tube, C_casing, A_fin;
        float Rho_Al, Rho_Copper, Rho_Metal, Rho_Solder;
        Rho_Al = 2700;                  //2700 is density of Aluminum
        Rho_Copper = 8800;
        Rho_Metal = 7700;
        Rho_Solder = 8800;
        A_fin = (S1*M) * ((N/M)*S1*0.866);
        C_fin = input.C_Aluminum * A_fin * (B/b) * delta * Rho_Al;
        C_tube = input.C_Copper * Rho_Copper * (3.14159 * (pow(d0,2) - pow((d0-
0.001),2)) /4) * (N*B);
        C_casing = input.C_Metal * Rho_Metal * 2 *
((S1*M+0.07)*(N*S1*0.866/M+0.07) + (B+0.04)*(N*S1*0.866/M)) * 0.0012;
        C_Material = C_fin + C_tube + C_casing;
        //Compute C_Weld
        double C_Weld;
        C_Weld = (57e-9) * Rho_Solder * input.C_Solder * 2*N;
        //Compute Cost
        Cost = 1.08*C_Material + C_Weld;
```

```
        pEvp->L = L;
        pEvp->B = B;
        pEvp->Y = Y;
        pEvp->Cost = Cost;
        pEvp->ETAf = ETAf;
        pEvp->Q0 = Q0;
        pEvp->Wfr = Wfr;
        pEvp->K0 = K0;

        return 1;
}


int CEvp::RecordDataIntoTable(EVP_DATA *pEvp, long SERIES)
{
        //Get Sequence Number
        lEvpNum = pSequence->NextVal("EVP");
        if (lEvpNum < 0) return -1;


        pEvpSet->m_strSort = " NUMBER ASC ";
        pEvpSet->Open();

        /*
        if (dlg.DoModal() == IDPAUSE)
        {
                pEvpSet->Close();
                delete pEvpSet;
                return 0;
        }*/

        pEvpSet->m_pDatabase->BeginTrans();
        TRY
        {
                pEvpSet->AddNew();

                pEvpSet->m_NUMBER = lEvpNum;
                pEvpSet->m_SERIES = SERIES;
                pEvpSet->m_DATA_VALID = pEvp->DATA_VALID;
                pEvpSet->m_S1       = pEvp->S1;
                pEvpSet->m_b        = pEvp->b;
                pEvpSet->m_Vac      = pEvp->Vac;
                pEvpSet->m_t2       = pEvp->t2;
                pEvpSet->m_Vfr      = pEvp->Vfr;
                pEvpSet->m_L        = pEvp->L;
```

```
                pEvpSet->m_B_WIDTH = pEvp->B;
                pEvpSet->m_Y       = pEvp->Y;
                pEvpSet->m_Cost    = pEvp->Cost;
                pEvpSet->m_Etaf = pEvp->ETAf;
                pEvpSet->m_Q0      = pEvp->Q0;
                pEvpSet->m_Wfr     = pEvp->Wfr;
                pEvpSet->m_K0      = pEvp->K0;

                pEvpSet->Update();
                pEvpSet->m_pDatabase->CommitTrans();
        }
        CATCH (CDBException, e)
        {
                pEvpSet->m_pDatabase->Rollback();
                ::AfxMessageBox("Can not add data to Database, trasaction rollback!",
MB_OK);
        }
        END_CATCH

        pEvpSet->Close();

        return 1;
}

//Record MEAN, STD, d-, d+ into Table
int CEvp::RecordMeanSTDdIntoTable(struct OUTPUT *pOutput, long SERIES)
{
        //Get Sequence Number
        long lNum;
        lNum = pSequence->NextVal("Mean_STD_d");
        //TRACE("EVP = %d\n", lEvpNum);
        if (lNum < 0) return -1;

        pSet->m_strSort = " NUMBER ASC ";
        pSet->Open();
        pSet->m_pDatabase->BeginTrans();
        TRY
        {
                pSet->AddNew();

                pSet->m_NUMBER  = lNum;
                pSet->m_SERIES          = SERIES;
                pSet->m_Mu_S1           = pOutput->Mu_S1;
                pSet->m_Mu_b            = pOutput->Mu_b;
                pSet->m_Mu_Vac          = pOutput->Mu_Vac;
                pSet->m_Mu_t2           = pOutput->Mu_t2;
```

```
                pSet->m_Mu_Vfr          = pOutput->Mu_Vfr;
                pSet->m_Mu_L            = pOutput->Mu_L;
                pSet->m_Mu_B_WIDTH      = pOutput->Mu_B;
                pSet->m_Mu_Y            = pOutput->Mu_Y;
                pSet->m_Mu_Cost         = pOutput->Mu_Cost;
                pSet->m_Mu_ETAf         = pOutput->Mu_ETAf;
                pSet->m_Mu_Q0           = pOutput->Mu_Q0;
                pSet->m_Mu_Wfr          = pOutput->Mu_Wfr;
                pSet->m_Mu_K0           = pOutput->Mu_K0;
                pSet->m_Sigma_S1        = pOutput->Sigma_S1;
                pSet->m_Sigma_b         = pOutput->Sigma_b;
                pSet->m_Sigma_Vac       = pOutput->Sigma_Vac;
                pSet->m_Sigma_t2        = pOutput->Sigma_t2;
                pSet->m_Sigma_Vfr       = pOutput->Sigma_Vfr;
                pSet->m_Sigma_L         = pOutput->Sigma_L;
                pSet->m_Sigma_B_WIDTH   = pOutput->Sigma_B;
                pSet->m_Sigma_Wfr       = pOutput->Sigma_Wfr;
                pSet->m_Sigma_Cost      = pOutput->Sigma_Cost;
                pSet->m_Sigma_ETAf      = pOutput->Sigma_ETAf;
                pSet->m_Sigma_Q0        = pOutput->Sigma_Q0;
                pSet->m_Sigma_K0        = pOutput->Sigma_K0;
                pSet->m_d1_NEG          = pOutput->d1_NEG;
                pSet->m_d1_POS          = pOutput->d1_POS;
                pSet->m_d2_NEG          = pOutput->d2_NEG;
                pSet->m_d2_POS          = pOutput->d2_POS;
                pSet->m_d3_NEG          = pOutput->d3_NEG;
                pSet->m_d3_POS          = pOutput->d3_POS;
                pSet->m_Z               = pOutput->Z;

                pSet->Update();
                pSet->m_pDatabase->CommitTrans();
        }
        CATCH (CDBException, e)
        {
                pSet->m_pDatabase->Rollback();
                ::AfxMessageBox("Can not add data to Database, trasaction rollback!",
MB_OK);
        }
        END_CATCH

        pSet->Close();

        return 1;
}
```

```
//Calculate Mean Value, Standard divaiation and d-, d+
int CEvp::Calc_MEAN_STD_d(int m, float IC)
{
        long i;
        long RecordNum;
        RecordNum = (input.b_num)*(input.Vac_num)*(input.t2_num)*(input.Vfr_num);

        struct OUTPUT output;
        output.Mu_S1        = 0;
        output.Mu_b         = 0;
        output.Mu_Vac       = 0;
        output.Mu_t2        = 0;
        output.Mu_Vfr       = 0;
        output.Mu_L         = 0;
        output.Mu_B         = 0;
        output.Mu_Y         = 0;
        output.Mu_Cost      = 0;
        output.Mu_ETAf      = 0;
        output.Mu_Q0        = 0;
        output.Mu_Wfr       = 0;
        output.Mu_K0        = 0;
        output.Sigma_S1     = 0;
        output.Sigma_b      = 0;
        output.Sigma_Vac    = 0;
        output.Sigma_t2     = 0;
        output.Sigma_Vfr    = 0;
        output.Sigma_L      = 0;
        output.Sigma_B      = 0;
        output.Sigma_Y      = 0;
        output.Sigma_Cost   = 0;
        output.Sigma_ETAf= 0;
        output.Sigma_Q0     = 0;
        output.Sigma_Wfr    = 0;
        output.Sigma_K0     = 0;
        output.d1_NEG       = 0;
        output.d1_POS       = 0;
        output.d2_NEG       = 0;
        output.d2_POS       = 0;
        output.d3_NEG       = 0;
        output.d3_POS       = 0;

        pEvpSet->m_strFilter.Format("%s%ld%s%ld", "NUMBER > ", lStart, " AND
NUMBER <= ", lEnd);
        pEvpSet->m_strSort = " NUMBER ASC ";
        pEvpSet->Open();
        if (!pEvpSet->IsEOF())
```

```
        {
                pEvpSet->MoveFirst();
        }
        //Get Mean
        while (!pEvpSet->IsEOF())
        {
                output.Mu_S1        += pEvpSet->m_S1;
                output.Mu_b         += pEvpSet->m_b;
                output.Mu_Vac       += pEvpSet->m_Vac;
                output.Mu_t2        += pEvpSet->m_t2;
                output.Mu_Vfr       += pEvpSet->m_Vfr;
                output.Mu_L         += pEvpSet->m_L;
                output.Mu_B         += pEvpSet->m_B_WIDTH;
                output.Mu_Y         += pEvpSet->m_Y;
                output.Mu_Cost      += pEvpSet->m_Cost;
                output.Mu_ETAf      += pEvpSet->m_Etaf;
                output.Mu_Q0        += pEvpSet->m_Q0;
                output.Mu_Wfr       += pEvpSet->m_Wfr;
                output.Mu_K0        += pEvpSet->m_K0;
                pEvpSet->MoveNext();
        }
        RecordNum           = pEvpSet->GetRecordCount();
        output.Mu_S1        = output.Mu_S1/RecordNum;
        output.Mu_b         = output.Mu_b/RecordNum;
        output.Mu_Vac       = output.Mu_Vac/RecordNum;
        output.Mu_t2        = output.Mu_t2/RecordNum;
        output.Mu_Vfr       = output.Mu_Vfr/RecordNum;
        output.Mu_L         = output.Mu_L/RecordNum;
        output.Mu_B         = output.Mu_B/RecordNum;
        output.Mu_Y         = output.Mu_Y/RecordNum;
        output.Mu_Cost      = output.Mu_Cost/RecordNum;
        output.Mu_ETAf      = output.Mu_ETAf/RecordNum;
        output.Mu_Q0        = output.Mu_Q0/RecordNum;
        output.Mu_Wfr       = output.Mu_Wfr/RecordNum;
        output.Mu_K0        = output.Mu_K0/RecordNum;

        //Get STD
        while( !pEvpSet->IsBOF( ) )
                pEvpSet->MovePrev( );
        pEvpSet->MoveFirst();
        while (!pEvpSet->IsEOF())
        {
                output.Sigma_S1             += (pEvpSet->m_S1 - output.Mu_S1) *
(pEvpSet->m_S1 - output.Mu_S1);
                output.Sigma_b              += (pEvpSet->m_b - output.Mu_b) *
(pEvpSet->m_b - output.Mu_b);
```

```
            output.Sigma_Vac        += (pEvpSet->m_Vac - output.Mu_Vac) *
(pEvpSet->m_Vac - output.Mu_Vac);
            output.Sigma_t2         += (pEvpSet->m_t2 - output.Mu_t2) *
(pEvpSet->m_t2 - output.Mu_t2);
            output.Sigma_Vfr        += (pEvpSet->m_Vfr - output.Mu_Vfr) *
(pEvpSet->m_Vfr - output.Mu_Vfr);
            output.Sigma_L          += (pEvpSet->m_L - output.Mu_L) *
(pEvpSet->m_L - output.Mu_L);
            output.Sigma_B          += (pEvpSet->m_B_WIDTH -
output.Mu_B) * (pEvpSet->m_B_WIDTH - output.Mu_B);
            output.Sigma_Y          += (pEvpSet->m_Y - output.Mu_Y) *
(pEvpSet->m_Y - output.Mu_Y);
            output.Sigma_Cost       += (pEvpSet->m_Cost - output.Mu_Cost) *
(pEvpSet->m_Cost - output.Mu_Cost);
            output.Sigma_ETAf  += (pEvpSet->m_Etaf - output.Mu_ETAf) *
(pEvpSet->m_Etaf - output.Mu_ETAf);
            output.Sigma_Q0         += (pEvpSet->m_Q0 - output.Mu_Q0) *
(pEvpSet->m_Q0 - output.Mu_Q0);
            output.Sigma_Wfr        += (pEvpSet->m_Wfr - output.Mu_Wfr) *
(pEvpSet->m_Wfr - output.Mu_Wfr);
            output.Sigma_K0         += (pEvpSet->m_K0 - output.Mu_K0) *
(pEvpSet->m_K0 - output.Mu_K0);
            pEvpSet->MoveNext();
        }
        output.Sigma_S1     = sqrt(output.Sigma_S1/(RecordNum-1));
        output.Sigma_b      = sqrt(output.Sigma_b/(RecordNum-1));
        output.Sigma_Vac    = sqrt(output.Sigma_Vac/(RecordNum-1));
        output.Sigma_t2     = sqrt(output.Sigma_t2/(RecordNum-1));
        output.Sigma_Vfr    = sqrt(output.Sigma_Vfr/(RecordNum-1));
        output.Sigma_L      = sqrt(output.Sigma_L/(RecordNum-1));
        output.Sigma_B      = sqrt(output.Sigma_B/(RecordNum-1));
        output.Sigma_Y      = sqrt(output.Sigma_Y/(RecordNum-1));
        output.Sigma_Cost   = sqrt(output.Sigma_Cost/(RecordNum-1));
        output.Sigma_ETAf   = sqrt(output.Sigma_ETAf/(RecordNum-1));
        output.Sigma_Q0     = sqrt(output.Sigma_Q0/(RecordNum-1));
        output.Sigma_Wfr    = sqrt(output.Sigma_Wfr/(RecordNum-1));
        output.Sigma_K0     = sqrt(output.Sigma_K0/(RecordNum-1));

        //Get d1-, d1+, d2-, d2+, d3-, d3+
        //(di-)*(di+) = 0;  di-,di+ >= 0;  1 =< i <= 3
        //System goal: Mu_Cost/IC + d1_NEG - d1_POS = 1
        output.d1_NEG = 0;
        output.d1_POS = output.Mu_Cost/IC - 1;
        //System goal: Mu_ETAf/0.8 + d2_NEG - d2_POS = 1
        output.d2_NEG = output.Mu_ETAf/0.8 - 1;
        output.d2_POS = 0;
```

```
        //Commonality goal: (Sigma_S1+Sigma_b+Sigma_Vac+Sigma_Vfr+Sigma_t2)/5
+ d3_NEG - d3_POS = 0
        output.d3_NEG = 0;
        output.d3_POS = (output.Sigma_S1 + output.Sigma_b + output.Sigma_Vac +
output.Sigma_L + output.Sigma_B)/5;

        //Get Z;
        //Z = w1*(d1_NEG+d1_POS) + w2*(d2_NEG+d2_POS) +
w3*(d3_NEG+d3_POS)    w1=w2=w3=1/3
        //Minimize Z
        float w1, w2, w3;
        w1 = 0.333;
        w2 = 0.333;
        w3 = 0.333;
        output.Z = w1*(output.d1_NEG+output.d1_POS) +
w2*(output.d2_NEG+output.d2_POS) + w3*(output.d3_NEG+output.d3_POS);

        RecordMeanSTDdIntoTable(&output, m+1);

        pEvpSet->Close();

        return 1;
}

//Calculate min Z
int CEvp::Calc_Z_min(struct OUTPUT *pResult)
{
        int begin, end;
        begin = 1;
        end = begin + input.S1_num;
        if (end = begin)        end++;

        pSet->m_strFilter.Format("%s %ld %s %ld", "NUMBER >= ", begin, " AND
NUMBER <= ", end);
        pSet->m_strSort = " Z ASC ";
        pSet->Open();

        pSet->MoveFirst();

        pResult->Mu_S1          = pSet->m_Mu_S1;
        pResult->Mu_b           = pSet->m_Mu_b;
        pResult->Mu_Vac         = pSet->m_Mu_Vac;
        pResult->Mu_t2          = pSet->m_Mu_t2;
        pResult->Mu_Vfr         = pSet->m_Mu_Vfr;
        pResult->Mu_L           = pSet->m_Mu_L;
        pResult->Mu_B           = pSet->m_Mu_B_WIDTH;
```

```
        pResult->Mu_Y           = pSet->m_Mu_Y;
        pResult->Mu_Cost        = pSet->m_Mu_Cost;
        pResult->Mu_ETAf        = pSet->m_Mu_ETAf;
        pResult->Mu_Q0          = pSet->m_Mu_Q0;
        pResult->Mu_Wfr         = pSet->m_Mu_Wfr;
        pResult->Mu_K0          = pSet->m_Mu_K0;
        pResult->Sigma_S1       = pSet->m_Sigma_S1;
        pResult->Sigma_b        = pSet->m_Sigma_b;
        pResult->Sigma_Vac      = pSet->m_Sigma_Vac;
        pResult->Sigma_t2       = pSet->m_Sigma_t2;
        pResult->Sigma_Vfr      = pSet->m_Sigma_Vfr;
        pResult->Sigma_L        = pSet->m_Sigma_L;
        pResult->Sigma_B        = pSet->m_Sigma_B_WIDTH;
        pResult->Sigma_Y        = pSet->m_Sigma_Y;
        pResult->Sigma_Cost     = pSet->m_Sigma_Cost;
        pResult->Sigma_ETAf     = pSet->m_Sigma_ETAf;
        pResult->Sigma_Q0       = pSet->m_Sigma_Q0;
        pResult->Sigma_Wfr      = pSet->m_Sigma_Wfr;
        pResult->Sigma_K0       = pSet->m_Sigma_K0;
        pResult->d1_NEG         = pSet->m_d1_NEG;
        pResult->d1_POS         = pSet->m_d1_POS;
        pResult->d2_NEG         = pSet->m_d2_NEG;
        pResult->d2_POS         = pSet->m_d2_POS;
        pResult->d3_NEG         = pSet->m_d3_NEG;
        pResult->d3_POS         = pSet->m_d3_POS;

        pSet->Close();

        return 1;
}


int CEvp::Process(struct OUTPUT *pResult)
{
        long m, n, p, q, r;
        int rtn;

        long    EvpNum;
        EvpNum =
(input.S1_num)*(input.b_num)*(input.Vac_num)*(input.t2_num)*(input.Vfr_num);
        //EvpNum = 100000;
        struct EVP_DATA temp;
        temp.IC = 0;


        long i = 0;
```

```
lStart = 0;
lEnd = 0;

temp.S1 = input.S1_min;
for(m = 0; m < input.S1_num; m++)
{
        lStart = lEnd ;
        temp.b  = input.b_min;
        for (n = 0; n < input.b_num; n++)
        {
                temp.Vac= input.Vac_min;
                for (p = 0; p < input.Vac_num; p++)
                {
                        temp.t2 = input.t2_min;
                        for (q = 0; q < input.t2_num; q++)
                        {
                                temp.Vfr= input.Vfr_min;
                                for (r = 0; r < input.Vfr_num; r++)
                                {
                                        //Calculate L, B, Cost, Etaf, Q0, K0
                                        rtn = Calculate(&temp);
                                        if (rtn != 1) return 0;
                                        //Compute IC
                                        if (i == 0)
                                        {
                                                temp.IC = temp.Cost;
                                        }
                                        else
                                        {
                                                if (temp.Cost < temp.IC)
                                                        temp.IC = temp.Cost;
                                        }

                                        //Judge group of data is valid or not
                                        rtn = JudgeFinEfficiency(temp.ETAf);
                                        if (rtn != 1)
                                        {
                                                temp.DATA_VALID = 0;
                                        }
                                        else
                                        {
                                                temp.DATA_VALID = 1;
                                        }

                                        //Record current group of data into table
                                        if (temp.DATA_VALID == 1)
```

```
                                        {
                                                rtn = RecordDataIntoTable(&temp,
m+1);
                                        }

                                        i++;
                                        temp.Vfr += input.Vfr_step;
                                }
                                temp.t2 += input.t2_step;
                        }
                        temp.Vac += input.Vac_step;
                }
                //if (t == 11)
                //      Calc_MEAN_STD_d(m, temp.IC);

                temp.b += input.b_step;
        }

        lEnd = lEvpNum;
        //Calculate Mean Value, STD, and d-, d+
        //if (t == 1)
        if (lEvpNum > 1 && lEnd > (lStart + 1))
        {
                Calc_MEAN_STD_d(m, temp.IC);
        }

        temp.S1 += input.S1_step;
    }

    Calc_Z_min(pResult);

    return 1;
}
```

VITA AUCTORIS

NAME: Caiyun Wang

PLACE OF BIRTH: Hunan, China

YEAR OF BIRTH: 1973

EDUCATION: Bachelor degree in Mechanical Engineering
Major: Heating, Ventilation and Air-conditioning Engineering
South Central University, China, 1996

Master of Applied Science in Industrial Engineering
University of Windsor
Windsor, ON, Canada, 2006