POSITION PAPER
OPTIMIZATION IN DECISION-BASED DESIGN

Farrokh Mistree and Janet K. Allen
Systems Realization Laboratory
The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0405

GLOSSARY

Designing[1] Designing is a process of converting information that characterizes the needs and requirements for a product into knowledge about a product. Thus designing becomes an issue of information processing.

Meta-Design[1] A metalevel process of designing systems that includes partitioning the system for function, partitioning the design process into a set of decisions and planning the sequence in which these decisions will be made.

Decision:[2] "...we define a decision as an irrevocable allocation of resources." "There are two important characteristics of a decision:

A *decision* is made at an instant in time.

A *decision* must be made based on the information available at the time it is made."

Information[3]. "The degrees of freedom that exist in a given situation to choose among signal symbols, messages, or patterns to be transmitted."

Knowledge The sum or range of what has been percieved, discovered and learned about a product.

Decision-Based Design:[1,4] The principal role of a designer, in Decision-Based Design, is to make decisions. Decisions help bridge the gap between an idea and reality. In Decision-Based Design, decisions serve as markers to identify the progression of a design from initiation to implementation to termination. In Decision-Based Design decisions represent a unit of communication; one that has both domain-dependent and domain-independent features.

Satisficing - not the "best" but "good enough". The first use of this term, in the context of optimization, is attributed to Herbert Simon[5].

ABSTRACT

We believe that design is decision-based and the role of "optimization" is to support human decision making. In this context, in this paper, we introduce and discuss decision models to support human decision making in design.

OUR FRAME OF REFERENCE

In this paper we explicate our views on Decision-Based Design. Since we are explicating our views we have taken the liberty of referencing our own work. In the papers we reference in this paper we extensively reference the work of others.

Our work is anchored in the works of Herbert Simon and James Miller. Simon, suggests in his book[5] that design is decision-based and one of the sciences of the artificial. The development of any science, particularly a science of the artificial, is anchored on a body of beliefs, hypotheses and knowledge. In our case, this anchor is provided by James Miller and his exposition of Living Systems Theory[3].

Decision-Based Design is a term coined to emphasize a different perspective from which to develop methods for design[1]. The principal role of a designer, in Decision-Based Design (DBD), is to make decisions. Decisions help bridge the gap between an idea and reality. In DBD, decisions serve as markers to identify the progression of a design from initiation to implementation to termination. Decisions represent a unit of communication; one that has both domain-dependent and domain-independent features. By focusing upon decisions a description of the processes written in a common "language" for teams from the various disciplines, a language that can be used in the process of designing, is obtained.

In this definition, the term product is used in its most general sense; it may include processes as well. The conversion of information into knowledge takes place through a process of decision making. The characteristics of design decisions are governed by the characteristics associated with the design of real-life engineering systems. These characteristics are summarized by the following descriptive sentences[1]:

- Decisions in design are invariably multileveled and multidimensional in nature.
- Decisions involve information that comes from different sources and disciplines.
- Decisions are governed by multiple measures of merit and performance.
- All the information required to make a decision may not be available.
- Some of the information used in making a decision may be hard, that is, based on scientific principles and some information may be soft, that is, based on the designer's judgment and experience.
- The problem for which a decision is being made is invariably loosely defined and open and is characterized by the lack of a singular, unique solution. The decisions are less than optimal and represent *satisficing* solutions.

Version: February 10, 1997.
Position Paper: *Optimization in Industry*, Palm Coast, Florida, March 23-27, 1997.
Position Paper: *DBD Workshop*, Orlando, Florida, April 1997.

These characteristics are equally applicable to decisions in other processes in the product life-cycle. Concurrent Design, in our view, is that part of Concurrent Engineering that is directly concerned with defining the product for manufacture and other downstream life-cycle processes. Any scheme for accomplishing concurrent design will have to take the above-mentioned decision characteristics into account. Clearly, in concurrent design, the product developers need to focus on making decisions concerning the system and related processes simultaneously and in so doing consider the interaction of these decisions.

In our opinion, there are three issues that bear scrutiny, namely, the creation of a decision-based model for integrated process and product development, the modeling and the evolution of decisions along a time-line and the issue of scaling. In this position paper we limit our remarks to the second issue.

THE DESIGN EQUATION

In order to identify a fundamental model for designing, we look to our paradigm of Decision-Based Design. In keeping with the definition of designing as a process of converting information that characterizes the needs and requirements for a product into knowledge about a product, we state our perception of knowledge, information and data. It seems that a preferred order has been developed among these three terms. Knowledge is the most concrete form. By reasoning, discussion and other mind-involving processes, knowledge is derived by human beings from information. Thus, information is always required *before* we obtain knowledge. Knowledge is *specific* information. The implication of this is that general information exists before knowledge (specific information) along a time-line. For instance, in science we use knowledge generated by other scientists as information in order to obtain new knowledge. Data is the simplest form and is characterized by a sense of hardness. Data and facts are often considered to be equal. Data is information, but not all information is data.

Using the preceeding connotations for information and knowledge, we are able to derive algebraic expressions for Decision-Based Design, namely, the Decision-Based Design equations. From our definition of designing we derive an algebraic representation for a single conversion of information into knowledge, namely, the *design equation*[6] ,

$$\mathbf{K} = \mathbf{T}(\mathbf{I}), \qquad (1)$$

where

- **I** is a vector with *n* components representing the information,
- **K** is a vector with *m* components representing the knowledge,
- **T**( ) is a function to transform the vector **I** into vector **K**; the transformation function **T**( ) comprises of a set of *m* functions, that is, $\mathbf{T}(\,) = (T_1(\,), T_2(\,), \ldots, T_m(\,))$.

The preceding equation represents a single transformation of information into knowledge. In Equation 1, **I** represents the information that will be transformed into the knowledge **K**. In the design equation, the transformation of information into knowledge is represented as a vector function, namely, **T**(). The design equation is the most general algebraic representation of a conversion of information into knowledge. In Figure 1, we present a graphical representation of the design equation with some practical examples of transformations of information into knowledge.

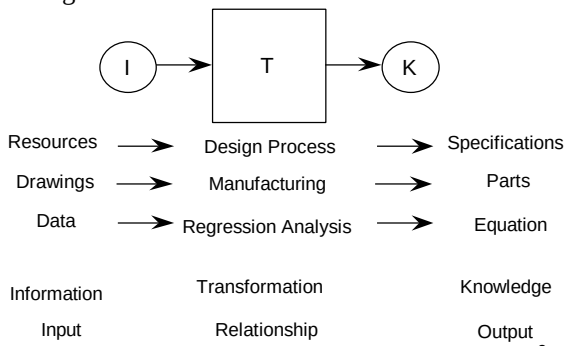

FIGURE 1: The Design Equation K=T(I) [6]

It is clear from Figure 1 that the design equation can take many forms. For instance, **K** may contain knowledge about a set of parts to be manufactured and **I** a set of drawings. Then **T**( ) represents the manufacturing process. In this case, we have captured the "design for manufacture" in our equation [6].

We model the design equation **K**=**T**(**I**) as a relationship. Relationships in a process can occur in different forms, for example, as a computer program, knowledge base, rule, neural network, mathematical programming formulation, a simple formula, etc. When knowledge and information take on different forms, a uniform information representation scheme becomes a necessity. However, a uniform representation scheme can only be developed by recognizing that all relationships require some form of input and generate an appropriate output. This view facilitates the combination of different types of relationships into networks and hierarchies. The hierarchies represent the knowledge and information held and generated at different levels of abstraction whereas the networks represent the change in knowledge and information along a time-line.

As an approximation of the design equation we introduce the *meta-design equation* [6],

$$\mathbf{DK} = [\mathbf{T}]\,\mathbf{DI}, \qquad (2)$$

where

DI is a vector with *n* components representing a difference in information,

DK is a vector with *m* components representing a difference in knowledge,

[**T**] is a *m x n* matrix transforming vector **DI** into vector **DK**.

The meta-design equation (Equation 2) is a special and simpler case of the design equation. The meta-design equation is a first order Taylor series expansion of the design equation. In the meta-design equation the conversion of information into knowledge is embodied in the transformation matrix [**T**]. We use the meta-design equation to convert a difference in information into a difference in knowledge due to a difference in information. In the meta-design equation we are concerned with designing a design process.

In the design equation (Equation 1), a single knowledge element K_i is expressed in terms of the information as [6]

$$\mathbf{K_i} = \mathbf{T_i(I)}. \quad (3)$$

In the meta-design equation (Equation 2), a single knowledge element K_i is expressed in terms of the information as [6]

$$\mathbf{DK_i} = \sum_{j=1}^{m} T_{ij}\Delta I_j \ . \quad (4)$$

The elements of the transformation matrix [**T**] in the meta-design equation (Equation 2) can be derived from the design equation, namely,

$$\mathbf{T}_{ij} = \frac{\partial(K_i)}{\partial(I_j)} = \frac{\partial(T_i(I))}{\partial(I_j)} \ . \quad (5)$$

One form of the matrix [**T**] in the meta-design equation is interpreted as equivalent to the matrix [**A**] in Suh's axiomatic design equation, **FR**=[**A**]**DP,** where FR is a vector of Functional Requirements and DP is a vector of Design Parameters [6]. However, the [**T**] matrix is not limited to functioning as an approximation between FRs and DPs. It has the capability to provide an approximate relationship between ΔK and ΔI for any design. The function $T_i()$ in Equation 1 is satisfied by multiple Decision-Support Problems (DSPs). Hence, DSPs are the implementation of the design equation within DBD.

DECISION-SUPPORT PROBLEMS

The Decision Support Problem (DSP) Technique is one embodiment of DBD. The DSP Technique consists of three principal components: a design philosophy rooted in systems thinking, an approach for identifying and formulating Decision Support Problems (DSPs), and software. The DSP Technique requires that a designer implement two phases, namely, a meta-design phase and a computer-based design phase. Meta-design is accomplished through *partitioning* a problem into its elemental DSPs and then devising a *plan* of action.

Decision Support Problems provide a means for modeling decisions encountered in design and the domain specific mathematical models so built are called *templates*. Multiple objectives, quantified using analysis-based "hard" and insight-based "soft" information, can be modeled in the DSPs. For real-world, practical systems, all of the information for modeling systems comprehensively and accurately in the early stages of the project may not be available. Therefore, the solution to the problem, even if one is obtained using optimization techniques, cannot be optimum with respect to the real world due to the inherent approximations in the model. However, this solution can be used to support a designer's quest for a superior solution. In a computer-assisted environment this support is provided in the form of optimal solutions for DSPs. Formulation and solution of DSPs provide a means for making the following types of decisions:

- **Selection** - making a choice between a number of possibilities taking into account a number of measures of merit or attributes[7,8,9].
- **Compromise** - a primary DSP - the determination of the "right" values (or combination) of design variables to describe the best satisficing system design with respect to constraints and multiple goals.[10].
- **Derived DSPs** (see Figure 2) - a combination of primary DSPs to model a complex decision, e.g., selection/selection, compromise/compromise and selection/compromise decisions[11,12,13,14].

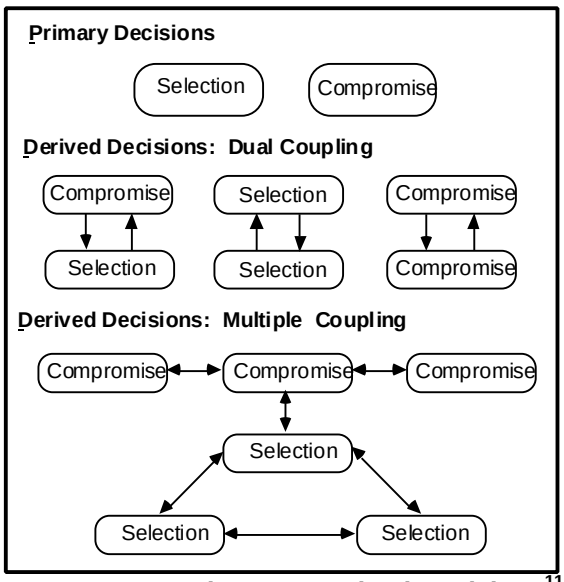

FIGURE 2 - Primary & Derived Decisions[11]

Selection and Compromise Decision Support Problems may be solved in an environment in which uncertainty

and risk are taken into consideration[15,16,17,18] Support is also provided for heuristic decisions which are made on the basis of a knowledge base of facts and rules of thumb; heuristic DSPs are solved using reasoning and logic only[19].

Some DBD-related publications: A decision-based taxonomy for multidisciplinary design is described by Lewis and Mistree in [20] and the modeling of interactions between disciplines using a game theoretic approach in [21]. A robust, decision-based method for concept exploration is documented by Chen and co-authors in [22,23]. The concept exploration method has been used in determining a ranged set of specifications for a product family[24]. A paradigm shift in design for manufacture that is rooted in DBD and living systems is proposed by Peplinski and co-authors in [25]. The processes of design, manufacture, and maintenance are modeled in the DSP Technique using entities, for example, phases, events, tasks, decisions, and information. A designer working within the DSP Technique has the freedom to use sub-models of a design process (prescriptive models) created and stored by others and also to create models (descriptive models) of their intended plan of action using the aforementioned entities. A prototype of a Design Guidance System, based on the Decision Support Problem Technique is described in [26,27]. A Design Learning Simulator for learning how to design using the DSP Technique may be accessed at URL: http://www.srl.gatech.edu/DLS

SATISFICING & OPTIMIZING

In the DBD paradigm the role of optimization is to *support* human judgment *not* replace it. Our focus therefore is to share our observations with respect to modeling such support for human decision making from the perspective of an *optimizer* and that of a *satisficer.* Let us explain. Consider a haystack with a number of needles hidden in it. An *optimizer* will continue searching the haystack until the last needle has been found. A *satisficer,* on the other hand, stops when he/she had found enough needles to proceed to the next step. We can capture the perspective of the optimizer by using the single objective optimization model and we can capture the perspective of the satisficer by using the compromise DSP. This has been explained in detail using a rotating disk example[28]. That leads us to the question: *What is the base-line model to capture a decision?*

We define, for the purposes of this paper, that the *baseline model* is the initial, unified crude mathematical description of a decision in decision-based design. It encompasses all classes of mathematical models. It is a vital phase of model development and consequent problem solving. A detailed description and examples of its use are given by Ignizio[29,30]. The omission of the baseline model can serve to both limit and distort a designer's concept of actual model development. At this point we make a distinction between constraints and goals in a design problem. *Constraints* are criteria that have to be met in order to obtain a feasible solution. *Goals* represent properties that are desirable in a design. In a real world problem, one can easily distinguish between the constraints and the goals. This understanding is very important and hence we make a clear distinction between the constraints and the goals in our baseline model. An interesting aspect of the baseline model representation is that it is independent of the approach used to solve the problem. A typical baseline model for a design problem is as follows:

Given An alternative that is to be improved through modification.
Assumptions used to model the domain of interest.
The system parameters.
All other relevant information including:

$A_i(\underline{X})$	achievement function i
G_i	goal or target value i
n	number of problem variables
t	number of goals
u	number of constraints
p	equality constraints
u-p	inequality constraints
r	maximizing objectives
s	minimizing objectives

Find The vector of problem variables $\underline{X}$

Satisfy The constraints

$$A_u(\underline{X}) \stackrel{=}{=} G_u \quad \text{for all u}$$

The goals

$$A_t(\underline{X}) \stackrel{=}{=} G_t \quad \text{for all t}$$

Maximize $A_r(\underline{X})$ for all r

Minimize $A_s(\underline{X})$ for all s.

Note that in this model we have **u** constraints, **t** goals, **r** maximizing objectives and **s** minimizing objectives.

Unfortunately, there is no optimization code available to solve the preceding baseline model in its most general form. Of course, one could develop some sort of heuristic scheme to solve all varieties of the baseline model and in the limit solve this problem by brute force. But, this is not germane to our current discussion.

Typically, we convert the base-line model to either a conventional single objective model or a multi-objective model that can be solved using an appropriate optimization algorithm. There are different modeling techniques to convert the baseline model for its solution by different codes. The philosophies behind the development of these modeling techniques vary and so they differ in distinct ways. The conversion of this

baseline model to the different modeling techniques, namely, the compromise DSP and the traditional single-objective approach is where our interest lies. The compromise DSP is a hybrid between goal programming[31] and mathematical programming models. Although, both goal programming and compromise DSPs share the concept of deviation variables (associated with deviations from target values for a designer's goals), what distinguishes them is the use of constraints and bounds in compromise DSPs.

TABLE 1 - The Optimization Problem versus the Compromise DSP

optimization problem		compromise DSP	
given	the feasible design space (consisting of constraints and bounds)	*given*	the feasible design space **and the aspiration space** (consisting of goals)
find	values for the system variables	*find*	the values for the system **and deviation** variables
satisfy	the feasible design space	*satisfy*	the feasible design space
optimize	a given objective function	*minimize*	**the discrepancy between the feasible design space and the aspiration space**

The optimization problem in Table 1 is stated as follows: ***given*** the feasible design space (consisting of constraints and bounds), ***find*** the values for the system variables that ***satisfy*** the feasible design space and ***optimize*** a given objective function. The statement for the compromise DSP in Table 1 reads: ***given*** the feasible design space and the aspiration space (consisting of goals), ***find*** the values for the system variables that ***satisfy*** the feasible design space and ***minimize*** the discrepancy between the feasible design space and the aspiration space**.** The difference in the two statements indicated as boldface text in Table 1 is of fundamental importance to the discussion that follows. The mathematical forms of a typical baseline model, a goal programming model, compromise DSP and the traditional single objective approach are given in Figure 3.

For the basis of comparison assume that there are **u** constraints, **t** goals, **r** maximizing objectives and **s** minimizing objectives in the baseline model. Of the **u** constraints, there are **p** equality constraints. The problem parameters and all other relevant information is given. This baseline model is then converted into the goal programming model [29-31], the compromise DSP[10], and the traditional single-objective model. The succinct form of these models is given in Figure 3. Note that for the traditional single objective model, we have to consider one

THE BASELINE MODEL	GOAL PROGRAMMING MODEL	THE COMPROMISE DSP	SINGLE-OBJECTIVE OPTIMIZATION MODEL
FIND The vector of problem variables ($\underline{X}$) **SATISFY** The **constraints** $A_u(\underline{X}) = G_u$ for all u The **goals** $A_t(\underline{X}) = G_t$ for all t **MAXIMIZE** $A_r(\underline{X})$ for all r **MINIMIZE** $A_s(\underline{X})$ for all s	**FIND** The vector of problem variables ($\underline{X}$)) The vector of deviation variables (d_i^-, d_i^+) **SATISFY** The **goals** $A_i(\underline{X}) + d_i^- - d_i^+ = G_i$ i = 1,..., u+r+s+t **MINIMIZE** Case a: Preemptive $Z = [f_1(d_i^-, d_i^+), ..., f_k(d_i^-, d_i^+)]$ Case b: Archimedean u+r+s+t $Z = \Sigma W_i(d_i^- + d_i^+)$; $\Sigma W_i = 1$; $W_i = 0$	**FIND** The vector of system variables ($\underline{X}$)) The vector of deviation variables (d_i^-, d_i^+) **SATISFY** The system **constraints** $g_i(\underline{X}) = 0$ i = 1, .., p $g_i(\underline{X}) = 0$ i = p+1,.,u The system **goals** $A_i(\underline{X}) + d_i^- - d_i^+ = G_i$ i = 1,..., r+s+t The **bounds** on the system $X_j^{min} = X_j = X_j^{max}$; j = 1,..., n $d_i^-, d_i^+ = 0$ and $d_i^- . d_i^+ = 0$ **MINIMIZE** Case a: Preemptive $Z = [f_1(d_i^-, d_i^+), ..., f_k(d_i^-, d_i^+)]$ Case b: Archimedean r+s+t $Z = \Sigma W_i(d_i^- + d_i^+)$; $\Sigma W_i = 1$; $W_i = 0$	**FIND** The vector of problem variables ($\underline{X}$)) **SATISFY** The **constraints** $g_i(\underline{X}) = 0$ i = 1, ..., p $g_i(\underline{X}) = G_i$ i = p+1, ...,u+r+s+t-1 **MINIMIZE** $A(\underline{X})$

FIGURE 3 - Models to Support Human Decision Making[8]

of the objectives or goals as the objective function and convert the rest of the objectives and goals into constraints. Therefore, in general, we would have to solve **u+r+s+t** different models. Of these **u+r+s+t** different solutions, the one that best "satisfies" all the objectives is chosen as the final solution. But then this may not be the answer, particularly if the problem has conflicting goals (objectives) and if the problem size is very large. The goal programming approach results in a *satisficing* solution to the decision problem. In this case, we have **u+r+s+t** goals and **2(u+r+s+t)** deviation variables. Therefore, the size of the problem is very large. Also we believe that in a real world there are always some constraints which we cannot violate and so if we disguise these constraints as goals then it does not represent the facts of the real world. This drawback is taken care of in the compromise DSP where we have **u** constraints and **r+s+t** goals. This results in **2(r+s+t)** deviation variables, which means that as compared to the goal programming approach, we have **2u** less deviation variables. It is emphasized that both the deviation variables, namely, underachievement (d-) and overachievement (d+) are always non-negative, with at least one of them being zero. These facts are taken into account by the two mathematical conditions $d_i^- , d_i^+ = 0$ and $d_i^- . d_i^+ = 0$. Also noted is the presence of bounds in the formulation of the compromise DSP. This is an advantage since bounds are a fact of life in engineering problems and hence they ought to be treated explicitly and differently from goals. We have substantiated the preceding discussion explicitly with an example, namely, the design of a rotating disk[28]. Other applications of DSPs include the design of ships, damage tolerant structural and mechanical systems, the design of aircraft, mechanisms, thermal energy systems, design using composite materials and data compression. A detailed set of references to these applications is presented in [1,32]. DSPs have been developed for hierarchical design: coupled selection-compromise, compromise-compromise and selection-selection. These constructs have been used to study interaction between design and manufacture[33] and between various events in the conceptual phase of the design process. The software for solving DSPs is called DSIDES (Decision Support in the Design of Engineering Systems) [10,34].

WHICH MODEL FOR SUPPORTING HUMAN JUDGMENT IN DBD?

In this section, we summarize our position that has resulted over several years of investigating Decision-Based Design and developing the Decision Support Problem Technique. We focus our attention on the two models, namely, the optimization model and the compromise DSP. Supporting arguments are presented in[28].

A Satisficer's Model: The Compromise DSP

- The availability of the preemptive and the Archimedean formulation enables a satisficer to model multiple objectives and the multi-level decisions as realistically as possible.
- The compromise DSP offers ease and flexibility in the modeling stage.
- The philosophy of the deviation function enables the designer to explore the design space comprehensively and develop further insight into the problem.
- The convergence of the deviation function to the same value from different starting points lends robustness to the compromise DSP.
- A limitation with the compromise DSP is that for the Archimedean formulation a designer has to depend on his/her experience and intuition to assign weights for the goals on the same preference level.
- Freedom to modify a given model includes the amount of deviation that can occur from a given target for all objectives and the type of the deviation function selected (Archimedean or preemptive).

An Optimizer's Model: The Optimization Model

- The traditional single-objective approach is advantageous when a designer is unaware of a target value for what he/she seeks. This, however, is of limited value when a designer seeks to model model-multi-level decisions.
- Since, a single objective drives the solution the issue of how this objective can and should be modeled is of major concern.
- True preemptive modeling of human preferences cannot be achieved.
- Multiple objectives occurring on different preference levels, involving a combination of preemptive and Archimedean formulations cannot be modeled.
- Freedom to modify a given model includes which of the multiple goals are to be converted to constraints and which goal is to be retained as the objective function.

Which model? The evolution of information along the design time-line, with the qualitative ratio of hard to soft information increasing, suggests that perhaps a satisficing model is appropriate for modeling decisions early on a product realization time-line with the optimization model being appropriate for the later stages[35,36]. The question then arises: When does one switch from a satisficing to an optimization model?

And now to some questions: What is the efficacy of approaching design from a decision-based perspective in an industrial setting? What is the role of optimization in Decision-Based Design? What are the models that support (rather than replace) human judgment? How do these models evolve along a design time-line? How can these models be verified? Why should design be developed as a science? And finally, what needs to be done to support practicing engineers design from a decision-based perspective?

REFERENCES

1. Mistree, F., Smith, W.F., Bras, B.A., Allen, J.K., and Muster, D. "Decision-Based Design: A Contemporary Paradigm for Ship Design," *Transactions of the Society of Naval Architects and Marine Engineers*, Vol. 98, 565-597, 1990.
2. Hazelrigg, G.A., *Engineering Systems*, Prentice Hall, Upper Saddle River, N.J., 1996.
3. Miller, J.G., Living Systems, McGraw-Hill, NY, 1978.
4. Shupe, J.A., *Decision-Based Design: Taxonomy and Implementation*, Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas, 1988.
5. Simon, H.A., *The Sciences of the Artificial*, The MIT Press, Cambridge, Massachusetts, 1982.
6. Bras, B., "Foundations for Designing Decision Based Design Processes," Ph.D. Dissertation, Cullen College of Engineering, University of Houston, Houston, Texas, 1992.
7. Kuppuraju, N., Ittimakin, P. and Mistree, F., "Design through Selection ... A Method that Works," *Design Studies*, Vol. 6, No. 2, 91-106, 1985.
8. Mistree, F., Marinopoulos, S., Jackson, D. and Shupe, J. A., "The Design of Aircraft using the Decision Support Problem Technique," NASA Contractor Report 4134, 1988.
9. Vadde, S., Allen, J. K. and Mistree, F., "Catalog Design: Selection using Available Assets," *Engineering Optimization*, Vol. 25, 45-64, 1995.
10. Mistree, F., Hughes, O.F., and Bras, B.A., "The Compromise Decision Support Problem and the Adaptive Linear Programming Algorithm," In *Structural Optimization: Status and Promise*, (M.P. Kamat, Editor) 247-286, AIAA, Washington, D.C., 1993.
11. Karandikar, H. M. and Mistree, F., "Modeling Concurrency in the Design of Composite Structures" in *Structural Optimization: Status and Promise*, 769-806, (M. P. Kamat, Ed.), Washington, D.C. AIAA, 1993.
12. Vadde, S., Allen, J.K. and Mistree, F., "Compromise Decisions for Hierarchical Design Under Uncertainty," *Computers and Structures*, Vol. 52, 645-658, 1994.
13. Bascaran, E., Bannerot, R. B. and Mistree, F., "Hierarchical Selection Decision Support Problems in Conceptual Design," *Engineering Optimization*, Vol. 14, 207-238, 1989.
14. Mistree, F., Smith, W.F. and Bras, B.A., "A

Decision-Based Approach to Concurrent Engineering," in *Handbook of Concurrent Engineering*, (H.R. Parsaei and W. Sullivan, Editors) New York, Chapman & Hall, 127-158, 1993.

[15] Allen, J.K., "The Decision to Introduce New Technology: The Fuzzy Preliminary Selection Decision Support Problem," *Engineering Optimization*, Vol. 26, No. 1, 61-77, 1996.

[16] Zhou, Q-J., Allen, J.K. and Mistree, F., "Decisions Under Uncertainty: The Fuzzy Compromise Decision Support Problem," *Engineering Optimization*, Vol. 20, 21-43, 1992.

[17] Reddy, R., and Mistree, F., "Modeling Uncertainty in Selection Using Exact Interval Arithmetic" in *Design Theory and Methodology 92*, (L.A. Stauffer and D.E. Taylor, Editors), ASME, 193-201, 1992.

[18] Vadde, S., Krishnamachari, R.S., Allen, J.K., and Mistree, F., "The Bayesian Compromise Decision Support Problems for Hierarchical Design Involving Uncertainty," ASME Journal of Mechanical Design, Vol. 116, 388-395, 1993.

[19] Kamal, S. Z., "The Development of Heuristic Decision Support Problems for Adaptive Design," Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas, 1990.

[20] Lewis, K. and Mistree, F., "On Developing a Taxonomy for Multidisciplinary Design Optimization: A Decision-Based Perspective," First World Congress of Structural and Multidisciplinary Optimization, 811-818, (N. Olhoff, G.I.N. Rozvany, eds.), Elsevier Science, Oxford, UK: ISSMO, 1995.

[21] Lewis, K. and Mistree, F., "Modeling Interactions in Multidisciplinary Design: A Game Theoretic Approach," AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, Washington, September 4-6, 755-765. AIAA Paper Number 96-4060, 1996.

[22] Chen, W., Allen, J.K., Mavris, D., Mistree, F, "A Concept Exploration Method for Determining Robust Top-Level Specifications," *Engineering Optimization*, Vol. 26, 137-158, 1996.

[23] Chen, W., Allen, J.K, Tsui, K-L and Mistree, F., "A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors," *ASME Journal of Mechanical Design*, Vol. 118, No. 4, 478-485, 1996.

[24] Simpson, T.W., Chen, W., Allen, J.K., and Mistree, F., "Conceptual Design of a Family of Products Through the Use of the Robust Concept Exploration Method," AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Bellevue, Washington, September 4-6, 1996, pp. 1535-1545. AIAA Paper Number 96-4161.

[25] Peplinski, J., Koch, P.N., Allen, J.K. and Mistree, F., "Design Using Available Assets: A Paradigm Shift in Design for Manufacture," *Concurrent Engineering: Research and Applications*, Vol. 4, No. 4, 317-332, 1996.

[26] Bras, B.A., Smith, W.F. and Mistree, F., "The Development of a Design Guidance Systems for the Early Stages of Design," in *CFD and CAD in Ship Design* (G. van Oortmerssen, Ed.) Wageningen, The Netherlands: Elsevier Science Publishers B.V., 221-231, 1990.

[27] Mistree, F., Bras, B. A., Smith, W. F. and Allen, J. K., "Modeling Design Processes: A Conceptual, Decision-Based Approach," *International Journal of Engineering Design and Automation*, Vol. 1, No. 4, 209-221, 1995.

[28] Mistree, F., Patel, B. and Vadde, S., "On Modeling Multiple Objectives and Multi-Level Decisions in Concurrent Design," Advances in Design Automation, (Gilmore, B.J., Hoeltzel, D., Dutta, D. and Eschenauer, H., Eds.), New York: ASME, 151-161. ASME DE-Vol. 69-2, 1994.

[29] Ignizio J. P., *Linear Programming in Single and Multi-Objective Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[30] Ignizio J. P., "Generalized Goal Programming: An Overview", *Computers and Operations Research*, Vol. 5, No. 3, 179-197, 1983.

[31] Ignizio J. P., *Introduction to Linear Goal Programming*, Quantitative Applications in the Social Sciences, (J. L. Sullivan and R. G. Niemi ed.), Sage University Papers, Beverly Hills, California, 1985.

[32] Mistree, F., Muster, D., Srinivasan, S. and Mudali, S., "Design of Linkages: A Conceptual Exercise in Designing for Concept," *Mechanism and Machine Theory*, Special Issue on "Theories of Design - Application to the Design of Machines," Vol. 25, No. 3, 273-286, 1990.

[33] Karandikar, H. M., "Hierarchical Decision Making for the Integration of Information from Design and Manufacturing Processes in Concurrent Engineering," Ph.D. Dissertation, Department of Mechanical Engineering, University of Houston, Houston, Texas, 1989.

[34] Lewis, K. and Mistree, F., "Foraging-Directed Adaptive Linear Programming: An Algorithm for Solving Nonlinear, Mixed, Discrete/Continuous Design Problems," Advances in Design Automation, (Dutta, D., Ed.), New York: ASME, ASME 96-DETC/DAC-1601, 1996.

[35] Vadde, S., Allen, J.K., Lucas, T. and Mistree, F., "On Modeling Design Evolution along a Design Time-Line," AIAA/NASA/USAF/ISSMO

Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, September 7-9, 1474-1482. Paper No. AIAA-94-4313-CP, 1994.

[36] Hale, M., Craig, J.I., Mistree, F. and Schrage, D., "DREAMS and IMAGE: A Model and Computer Implementation for Concurrent, Life-Cycle Design of Complex Systems," *Concurrent Engineering: Research and Applications,* Vol. 4, No. 2, 171-186, 1996.