Hierarchical Imprecise Design with Weights

William S. Law
Erik K. Antonsson
Engineering Design Research Laboratory
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, California

Abstract

An extension to the *method of imprecision* that allows the decomposition or aggregation of imprecise design problems with weighted attributes is presented. The method of imprecision uses the preferences of the designer and customer to define fuzzy sets that quantify the imprecision associated with a design attribute. These fuzzy sets, weighted by relative importance, are combined to produce an overall design measure. This paper introduces aggregation operators that can be hierarchically applied to combine weighted design attributes so that the structure of the design problem is appropriately modeled.

Introduction

Imprecision is an integral part of the engineering design process: not imprecision in thought or logic, but the intrinsic vagueness of an unfinished design description. Early in the design process, the design description is nearly completely vague or imprecise (fuzzy). As design decisions are made, this imprecision is reduced, until ultimately the final design description is precise (crisp). Designers need to evaluate designs early in the design process, before key decisions are made, and while the design embodies the highest levels of imprecision. Approximately 70% of the cost of a product is determined during the preliminary design phase (Ullman, 1992). Yet existing design methods and computer-aids ignore the imprecision that is an essential part of the preliminary design process.

The *method of imprecision* (Wood, Otto, and Antonsson, 1992; Otto, 1992) is a formal methodology, based on fuzzy sets, for representing and manipulating imprecise preliminary design information. Fuzzy sets provide a natural representation of imprecision. This paper summarizes the principles and practice of the method of imprecision, and describes how it may be extended to accommodate design problems with a hierarchical structure of weighted attributes.

Definitions and Notation

An imprecise variable is a variable that may potentially assume any value within a possible range because the designer does not know, *a priori*, the final value that will emerge from the design process. Yet even though the designer is uncertain about what value to specify, certain values will be preferred over others. This preference, which may arise objectively (*e.g.* , cost or availability of components) or subjectively (*e.g.* , from experience), is used to quantify the imprecision associated with a design variable. Thus the designer's experience and judgement are incorporated into the design evaluation.

Design variables are denoted d_i, where i ranges from 1 to n. The whole set of design variables for each design alternative is an n vector, $\vec{d}$, which is an element of the *design variable space* (DVS). The valid design variable values within the DVS form a subset $\mathcal{X}$. The set of valid values for d_i is denoted $\mathcal{X}_i$.

The preference that a designer has for values of

d_i, the ith design variable, is represented by a preference function on $\mathcal{X}$, termed the *design preference*:

$$\mu_{d_i}(d_i) : \mathcal{X}_i \rightarrow [0,1] \subset \mathbb{R}$$

where the $\mathcal{X}_i$ are assumed to be compact. $\mu_{d_i}(d_i)$ quantifies the designer's preference for values of d_i, and is distinct from the customary membership function in a fuzzy set, which quantifies the extent to which values belong to the set.

Performance variables are denoted p_j, where j ranges from 1 to q. Each performance variable p_j is defined by a mapping f_j such that $p_j = f_j(\vec{d})$. The mappings f_j can be any calculation or procedure to measure the performance of a design, including closed-form equations, iterative and heuristic methods, "black box" functions, experiments, and consumer evaluations. The set of performance variables for each design alternative is a q vector, $\vec{p} = \vec{f}(\vec{d})$, which is an element of the *performance variable space* (PVS). The subset of valid performance variable values $\mathcal{Y}$ is mapped from $\mathcal{X}$ and the set of valid values for p_j is denoted $\mathcal{Y}_j$.

The customer's preference for values of p_j, the jth performance variable, is represented by a preference function on $\mathcal{Y}$, termed the *functional requirement*:

$$\mu_{p_j}(p_j) : \mathcal{Y}_j \rightarrow [0,1] \subset \mathbb{R}.$$

The combined preference of the designer and customer for a particular design $\vec{d}$ is represented by an overall preference $\mu_o(\vec{d})$, which is a function of the designer preferences $\mu_{d_i}(d_i)$, and the functional requirements $\mu_{p_j}(p_j) = \mu_{p_j}(f_j(\vec{d}))$:

$$\mu_o = \mathcal{P}\left(\mu_{d_1}, ..., \mu_{d_n}, \mu_{p_1}, ..., \mu_{p_q}\right).$$

The design problem is to identify design configurations that maximize μ_o, *i.e.* , designs $\vec{d}^*$ such that:

$$\mu_o(\vec{d}^*) = \mu_o^* = \sup\{\mu_o(\vec{d}) \mid \vec{d} \in \mathcal{X}\}.$$

The peak overall preference in $\mathcal{X}$, μ_o^*, is equal to the peak overall preference in $\mathcal{Y}$ (Otto, Lewis, and Antonsson, 1993).

The *combination function* $\mathcal{P}$ must satisfy the following four axioms to be consistent with engineering design (Otto, 1992):

Monotonicity

$$\mathcal{P}(\mu_1, ..., \mu_k, ..., \mu_{p+q}) \leq \mathcal{P}(\mu_1, ..., \mu_k', ..., \mu_{p+q}) \quad \text{iff } \mu_k \leq \mu_k' \quad \forall k \tag{1}$$

Continuity

$$\mathcal{P}(\mu_1, ..., \mu_k, ..., \mu_{p+q}) = \lim_{\mu_k' \rightarrow \mu_k} \mathcal{P}(\mu_1, ..., \mu_k', ..., \mu_{p+q}) \quad \forall k \tag{2}$$

Annihilation

$$\mathcal{P}(\mu_1, ..., 0, ..., \mu_{p+q}) = 0 \tag{3}$$

Idempotency

$$\mathcal{P}(\mu, ..., \mu) = \mu. \tag{4}$$

$\mathcal{P}$ reflects the design or trade-off strategy, which indicates how competing attributes of the design should be traded-off against each other (Otto and Antonsson, 1991a, 1991b). The appropriate design strategy is usually dictated by the design problem or sub-problem and is not a choice the designer can freely make. A design problem will in general require a hierarchy of different trade-off strategies which successively aggregate design attributes.

Suppose that only the lowest preference (μ_{d_i} or μ_{p_j}) is to be considered in evaluating the design: higher preferences for other attributes of the design cannot compensate for a lower preference. This is a *non-compensating* design strategy for which the combination function is $\mathcal{P}_{\min}$:

$$\mu_o(\vec{d}) = \min\left(\mu_{d_1}, ..., \mu_{d_n}, \mu_{p_1}, ..., \mu_{p_q}\right). \tag{5}$$

Consider a system of components, where the failure of one component results in the failure of the system such that the entire assembly must be replaced. The preferences on the times to failure for the components should be combined using a non-compensating design strategy, since a high preference corresponding to a long time to failure cannot compensate for a low preference corresponding to a short time to failure.

Alternatively, different attributes of the design may be traded-off, so that a more acceptable attribute partially compensates for a less acceptable attribute. This is a *compensating* design strategy for which the combination function is $\mathcal{P}_{\prod}$:

$$\mu_o(\vec{d}) = \left(\prod_{i=1}^{n} \mu_{d_i} \prod_{j=1}^{q} \mu_{p_j}\right)^{\frac{1}{n+q}}. \tag{6}$$

For an ordinary household battery, the performance variables battery life (energy stored) and unit cost may be traded off using a compensating strategy. A low cost partially compensates for a short battery life; a long battery life partially compensates for a high cost.

μ_d	design preference (designer)
$\mu_d(\vec{d})$	design preference on the DVS
$\mu_d(\vec{p})$	design preference induced onto the PVS
μ_p	functional requirement (customer)
$\mu_p(\vec{p})$	functional requirement on the PVS
$\mu_p(\vec{d})$	functional requirement mapped back onto the DVS
μ_o	overall preference, $\mathcal{P}(\mu_d, \mu_p)$
$\mu_o(\vec{d})$	overall preference mapped onto the DVS
$\mu_o(\vec{p})$	overall preference mapped onto the PVS
μ_o^*	peak overall preference
$\mathcal{P}_{\min}$	non-compensating combination function
$\mathcal{P}_{\Pi}$	compensating combination function

The Method of Imprecision

After specifying design preferences μ_{d_i} on $\mathcal{X}_i$ and functional requirements $\mu_{p_j}(p_j)$ on $\mathcal{Y}_j$, and identifying the appropriate design strategy, the first step is to combine the individual μ_{d_i} to obtain μ_d, the combined design preference. μ_d combines only the design preferences and corresponds to a lower tier in the hierarchy of design trade-offs that comprise the combination function $\mathcal{P}$:

$$\begin{aligned}\mu_o &= \mathcal{P}_c\left[\mu_d, \mu_p\right] \\ &= \mathcal{P}_c\left[\mathcal{P}_d(\mu_{d_1}, \ldots, \mu_{d_n}), \mathcal{P}_p(\mu_{p_1}, \ldots, \mu_{p_q})\right] \quad (7)\end{aligned}$$

where $\mathcal{P}_d$ combines the design preferences, $\mathcal{P}_p$ combines the functional requirements, and $\mathcal{P}_c$ combines these sub-results. Note that Equation (7) applies on both the DVS and the PVS.

Next μ_{d_i} is induced onto $\mathcal{Y}$, using the extension principle (Zadeh, 1965):

$$\mu_d(\vec{p}) = \sup_{\vec{d}:\vec{p}=\vec{f}(\vec{d})} \left[\mu_d(\vec{d})\right]$$

where sup over the null set is defined to be zero. $\mu_d(\vec{d})$ is the combined design preference on $\mathcal{X}$, as distinct from $\mu_d(\vec{p})$, the combined design preference induced onto $\mathcal{Y}$. $\mu_d(\vec{p})$ is obtained by mapping $\mu_d(\vec{d})$ onto the PVS.

$\mu_d(\vec{p})$ is calculated using the *Level Interval Algorithm*, or *LIA* (Wood et al., 1992; Otto and Antonsson, 1991b), first proposed by Dong and Wong (Dong and Wong, 1987) as the "Fuzzy Weighted Average" algorithm and also called the "Vertex Method". $\mu_d(\vec{p})$ and $\mu_p(\vec{p})$ are then combined using $\mathcal{P}_c$ from Equation (7) to obtain $\mu_o(\vec{p})$, the overall preference on $\mathcal{Y}$. The set of peak preference performances $\mathcal{Y}^* = \{\vec{p}^* \in \mathcal{Y} \mid \mu_o(\vec{p}^*) = \mu_o^*\}$ is found from $\mu_o(\vec{p})$.

The design problem is to find the set of peak preference designs $\mathcal{X}^* = \{\vec{d}^* \in \mathcal{X} \mid \mu_o(\vec{d}^*) = \mu_o^*\}$. For a typical engineering design problem, the inverse mapping $\vec{f}^{-1} : \mathcal{Y} \to \mathcal{X}$ is unavailable, but $\mu_o(\vec{d})$ can still be obtained point by point (Law and Antonsson, 1994).

The method of imprecision is iterative. Preferences evolve through successive evaluations of imprecise design alternatives. Yet this is not traditional "point-by-point" design iteration: the methodology uses fuzzy sets representing preferences over many designs, providing more complete information earlier in the design process. This information can be propagated to downstream engineering groups, facilitating fuzzy set based concurrent design. Set-based concurrent design (Ward, Liker, Sobek, and Cristiano, 1994) is a powerful paradigm that is enhanced by the use of fuzzy sets.

Weights

In combining the designer's and customer's preferences, their relative importance must be considered. This is achieved by assigning individual *weights* to each variable:

$$\begin{aligned}0 \le \omega_{d_i} \le 1 \\ 0 \le \omega_{p_j} \le 1.\end{aligned}$$

Each weight may vary with the value of the variable that it is associated with and thus importance is a function of $\vec{d}$ and $\vec{p}$. This variation is assumed to be continuous, so that similar designs with similar performances will have similar sets of weighting functions. The difficulty of specifying weights can be reduced by allowing the process to be iterative. Weights are relative and should be normalized:

$$\sum_{i=1}^{n} \omega_{d_i} + \sum_{j=1}^{q} \omega_{p_j} = 1.$$

Other issues concerning weighting functions are discussed in (Otto, 1992).

With the addition of weights, the combination function for a non-compensating design strategy, $\mathcal{P}_{\min}$, becomes (Otto and Antonsson, 1991a):

$$\begin{aligned}\mu_o &= \left(\min\{\mu_{d_1}^{\omega_{d_1}}, \ldots, \mu_{d_n}^{\omega_{d_n}}, \mu_{p_1}^{\omega_{p_1}}, \ldots, \mu_{p_q}^{\omega_{p_q}}\}\right)^{\frac{1}{\omega_{\max}}} \\ \omega_{\max} &= \max\{\omega_{d_1}, \ldots, \omega_{d_n}, \omega_{p_1}, \ldots, \omega_{p_q}\}.\end{aligned} \quad (8)$$

Equation (8) reduces to Equation (5) when all vari-

ables have the same weight ($\omega_{d_i} = \omega_{p_j} = \frac{1}{p+q}$ for all i, j). With the addition of weights, the combination function for a compensating design strategy, $\mathcal{P}_\Pi$, becomes (Otto and Antonsson, 1991a):

$$\mu_o = \prod_{i=1}^{n} {\mu_{d_i}}^{\omega_{d_i}} \prod_{j=1}^{q} {\mu_{p_j}}^{\omega_{p_j}}. \tag{9}$$

Equation (9) reduces to Equation (6) when all variables have the same weight.

Hierarchical Weighted Design

Suppose that the $n+q$ design and performance variables are split into two subsets so that a different trade-off strategy can be applied to each:

$$\begin{aligned}\mu_o &= \mathcal{P}(\mu_1, ...\mu_{n+q}) \\ &= \mathcal{P}_c\left[\mathcal{P}_a(\mu_1, ..., \mu_k), \mathcal{P}_b(\mu_{k+1}, ..., \mu_{n+q})\right].\end{aligned} \tag{10}$$

This is a generalization of Equation (7). How should the subordinate combination functions $\mathcal{P}_a$ and $\mathcal{P}_b$ and the superordinate combination function $\mathcal{P}_c$ be defined? If all variables are combined using the same trade-off strategy, $\mathcal{P}$ is given by Equation (8) for a non-compensating strategy and Equation (9) for a compensating strategy. $\mathcal{P}_a$, $\mathcal{P}_b$, and $\mathcal{P}_c$ for each strategy must satisfy Equation (10) for these two cases as well as monotonicity, continuity, annihilation, and idempotency (*i.e.* Equations (1), (2), (3), and (4)). Yet these conditions do not uniquely specify $\mathcal{P}_a$, $\mathcal{P}_b$, and $\mathcal{P}_c$, and in particular, do not indicate how weights should be combined and interpreted.

Redefine the non-compensating combination function $\mathcal{P}_{\min}$ to accommodate a subset of variables:

$$\begin{aligned}\mathcal{P}_{\min}[\mu_1, ..., \mu_k] &= [\min({\mu_1}^{\omega_1}, ..., {\mu_k}^{\omega_k})]^{\frac{1}{\omega_{\max}}} \\ \omega_{\max} &= \max(\omega_1, ..., \omega_k).\end{aligned} \tag{11}$$

Note that the weights $\omega_1, ..., \omega_k$ do not form a complete set and need not sum to one. The hierarchical operation $\mathcal{P}_{\min}$ combines k individual preferences with their associated weights, into a single preference that also has an associated weight: $\omega_{\max}$. If $\mathcal{P}$, $\mathcal{P}_a$, $\mathcal{P}_b$, and $\mathcal{P}_c$ are specified as this hierarchical non-compensating function , then Equation (10) is satisfied (set $N = n + q$):

$$\begin{aligned}\omega_{\max} &= \max[\omega_1, ..., \omega_N] \\ &= \max[\max(\omega_1, ..., \omega_k), \max(\omega_{k+1}, ..., \omega_N)] \\ &= \max[\omega_{a_{\max}}, \omega_{b_{\max}}]\end{aligned} \tag{12}$$

$$\begin{aligned}&\mathcal{P}_c\left[\mathcal{P}_a(\mu_1, ..., \mu_k), \mathcal{P}_b(\mu_{k+1}, ..., \mu_N)\right] \\ &= \mathcal{P}_c\left[\min({\mu_1}^{\omega_1}, ..., {\mu_k}^{\omega_k})^{\frac{1}{\omega_{a_{\max}}}}, \right. \\ &\quad \left. \min({\mu_{k+1}}^{\omega_{k+1}}, ..., {\mu_N}^{\omega_N})^{\frac{1}{\omega_{b_{\max}}}}\right] \\ &= \min\left[\min({\mu_1}^{\omega_1}, ..., {\mu_k}^{\omega_k}), \right. \\ &\quad \left. \min({\mu_{k+1}}^{\omega_{k+1}}, ..., {\mu_N}^{\omega_N})\right]^{\frac{1}{\omega_{\max}}} \\ &= \min\left[{\mu_1}^{\omega_1}, ..., {\mu_N}^{\omega_N}\right]^{\frac{1}{\omega_{\max}}} \\ &= \mathcal{P}[\mu_1, ..., \mu_N].\end{aligned} \tag{13}$$

Now redefine the compensating combination function $\mathcal{P}_\Pi$:

$$\begin{aligned}\mathcal{P}_\Pi[\mu_1, ..., \mu_N] &= \left(\prod_{i=1}^{k} {\mu_i}^{\omega_i}\right)^{\frac{1}{\omega}} \\ \omega &= \sum_{i=1}^{k} \omega_i.\end{aligned} \tag{14}$$

The single preference obtained by combining the multiple preferences $\mu_1, ..., \mu_k$ with their associated weights, using the hierarchical operation $\mathcal{P}_\Pi$, has an associated weight ω that is the sum of the individual weights. If $\mathcal{P}$, $\mathcal{P}_a$, $\mathcal{P}_b$, and $\mathcal{P}_c$ are specified as this hierarchical compensating function, then Equation (10) is satisfied (set $N = n + q$):

$$\begin{aligned}\omega_c &= \sum_{i=1}^{N} \omega_i \\ &= \sum_{i=1}^{k} \omega_i + \sum_{j=k+1}^{N} \omega_j \\ &= \omega_a + \omega_b\end{aligned} \tag{15}$$

$$\begin{aligned}&\mathcal{P}_c\left[\mathcal{P}_a(\mu_1, ..., \mu_k), \mathcal{P}_b(\mu_{k+1}, ..., \mu_N)\right] \\ &= \mathcal{P}_c\left[\left(\prod_{i=1}^{k} {\mu_i}^{\omega_i}\right)^{\frac{1}{\omega_a}}, \left(\prod_{i=k+1}^{N} {\mu_j}^{\omega_j}\right)^{\frac{1}{\omega_b}}\right] \\ &= \left[\left(\prod_{i=1}^{k} {\mu_i}^{\omega_i}\right)\left(\prod_{i=k+1}^{N} {\mu_j}^{\omega_j}\right)\right]^{\frac{1}{\omega_a+\omega_b}} \\ &= \left[\prod_{i=1}^{N} {\mu_i}^{\omega_i}\right]^{\frac{1}{\omega_c}} \\ &= \mathcal{P}[\mu_1, ..., \mu_N].\end{aligned} \tag{16}$$

The hierarchical combination functions for the non-compensating and compensating trade-off strategies (Equations (11) and (14)) can be successively

applied, allowing multiple levels of problem decomposition or aggregation: N may be less than or greater than $n + q$. It has been shown above that these functions remain consistent between adjacent levels of aggregation. Since they are merely generalized forms of the min and product of powers combination functions in Equations (8) and (9), they also satisfy monotonicity, continuity, annihilation, and idempotency (Equations (1), (2), (3), and (4)).

For the compensating strategy, the weights for aggregated attributes are added. This is not an arbitrary choice: the form of the compensating trade-off function requires the addition of weights to preserve the consistency of the hierarchical decomposition. Decomposition does not change the relative importance of design attributes. But does an aggregated weight represent the combined importance of the aggregated attributes? Equations (15) and (16) show that, mathematically, it does. Intuitively, it is also clear how adding weights aggregates the importance of individual attributes. For a compensating strategy, all attributes are considered, and thus combining attributes should result in an aggregated attribute that is more important. Furthermore, the importance of the aggregated attribute should depend on the importance of every one of its constituent attributes.

The compensating combination function $\mathcal{P}_{\Pi}$ given by Equation (14) is self-normalizing: scaling the weights by any positive factor does not affect the result. Consequently, weights do not need to sum to one, and can be scaled whenever necessary. For example, when two design problems are aggregated, the two sets of weights should be scaled in order to correctly represent the relative importance of the two problems.

For the non-compensating strategy, the maximum of the individual weights associated with the individual attributes becomes the aggregated weight associated with the combined attributes. The form of the non-compensating trade-off function dictates that the maximum weight must represent the aggregated attributes, in order to preserve the consistency of the hierarchical decomposition. Equations (12) and (13) show that aggregating weights in this way is mathematically correct. Yet it is not immediately obvious how a maximum can represent the combined importance of the aggregated attributes. The confusion arises because a non-compensating strategy considers *only* the minimum weighted preference in evaluating the design: all other preferences are discarded. Therefore only the preference and weight associated with the critical attribute are propagated to the next higher level in the design hierarchy.

The non-compensating combination function $\mathcal{P}_{\min}$ in Equation (11) is also self-normalizing and hence weights can be scaled by any positive factor without affecting the result. Note that, because only the maximum weight emerges from an aggregation operation, normalized weights will no longer sum to one after they are aggregated. Aggregation preserves the relationship between weights, but not their sum.

Conclusion

The method of imprecision is a formal methodology for incorporating imprecision into the design process, that is consistent with engineering design principles. The method uses the preferences of the designer and customer to define fuzzy sets that quantify the imprecision associated with a design attribute. These fuzzy sets, weighted by relative importance, can be traded-off with either a compensating or non-compensating combination strategy to produce an overall design measure. Compared to traditional design methodologies, this approach provides more complete information, earlier in the design process: information that can be propagated to downstream engineering groups, facilitating fuzzy set based concurrent design.

The hierarchical combination functions presented in this paper allow either a single imprecise design problem to be successively decomposed into multiple levels of sub-problems, or multiple imprecise design problems to be aggregated into a single super-problem. Decomposition does not modify the design problem because the aggregation operation preserves the meaning and validity of aggregated preferences and weights. Non-compensating and compensating combination functions can be applied interchangeably to fit the structure of the problem and weights can be freely scaled in order to accurately represent the relative importance of a sub-problem.

Acknowledgments

This material is based upon work supported, in part, by: The National Science Foundation under a Presidential Young Investigator Award, Grant

No. DMC-8552695, and NSF Grant No. DDM-9201424. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

References

Dong, W. M., and Wong, F. S., 1987, "Fuzzy Weighted Averages and Implementation of the Extension Principle," *Fuzzy Sets and Systems*, Vol. 21, (No. 2), pp. 183–199.

Law, W. S., and Antonsson, E. K., 1994, "Implementing the Method of Imprecision: An Engineering Design Example," In *Proceedings of the Third IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '94)*, Vol. 1, pp. 358–363. IEEE Invited paper.

Otto, K. N., 1992, *A Formal Representational Theory for Engineering Design* Ph.D. thesis, California Institute of Technology, Pasadena, CA.

Otto, K. N., and Antonsson, E. K., 1991a, "Trade-Off Strategies in Engineering Design," *Research in Engineering Design*, Vol. 3, (No. 2), pp. 87–104.

Otto, K. N., and Antonsson, E. K., 1991b, "Trade-Off Strategies in the Solution of Imprecise Design Problems," In Terano, T., et al. eds., *Fuzzy Engineering toward Human Friendly Systems: Proceedings of the International Fuzzy Engineering Symposium '91, Volume 1*, pp. 422–433 Yokohama Japan. LIFE, IFES.

Otto, K. N., Lewis, A. D., and Antonsson, E. K., 1993, "Determining Optimal Points of Membership with Dependent Variables," *Fuzzy Sets and Systems*, Vol. 60, (No. 1), pp. 19–24.

Ullman, D. G., 1992, *The Mechanical Design Process* McGraw Hill, New York.

Ward, A. C., Liker, J. K., Sobek, D. K., and Cristiano, J. J., 1994, "Set-Based Concurrent Engineering and Toyota," In *Design Theory and Methology - DTM '94*, Vol. DE68, pp. 79–90. ASME.

Wood, K. L., Otto, K. N., and Antonsson, E. K., 1992, "Engineering Design Calculations with Fuzzy Parameters," *Fuzzy Sets and Systems*, Vol. 52, (No. 1), pp. 1–20.

Zadeh, L. A., 1965, "Fuzzy Sets," *Information and Control*, Vol. 8, pp. 338–353.