

A Decision Aid For Selecting Among Information System Alternatives

By: Gary Klein
Department of Management
Information Sciences
Edwin L. Cox School of Business
Southern Methodist University
Dallas, Texas

By: Philip O. Beck
American Airlines
MD 2B56
P.O. Box 619616
DFW Airport, Texas 75261

Abstract

Many information system selection decisions, software decisions, hardware decisions, and project selection decisions involve characteristics that are of a qualitative nature. Present methodologies either ignore the problem of selecting qualitative attributes, or require the assignment of a numerical value to a qualitative variable in order to make a choice involving qualitative benefits. These approaches require application of artificial measures to qualitative variables. A procedure that overcomes these difficulties for selection problems with known alternatives is presented. The procedure is an implicit enumeration procedure based on properties of preference theory, requires no formal optimization procedure, and would be simple to implement in an interactive decision setting involving selection from among a set of known alternatives.

Keywords: Project selection, multiple criteria decisions, system evaluation

ACM Categories: H.0, K.6.2, K.6.3

Introduction

Qualitative attributes are those attributes which can be identified but cannot be quantified in meaningful (numerical) terms. Qualitative attributes may be important elements in a decision, but the lack of a quantified value for them restricts their use in formal decision models. Color, for example, is a qualitative attribute used in many consumer decisions. Yet few consumers would be able to tell you a scale which represents their color selection. Color could be quantified in terms of wavelength characteristics, but to use this measure to decide the color of your next car would be ludicrous. A concern, then, is to find a technique which will enhance the decision maker's ability to absorb all relevant factors, including qualitative information.

Qualitative attributes are prevalent in the development of information systems. Hardware selection involves consideration of the quantitative cost, power, and capacity attributes along with qualitative service and ergonomic attributes. Software selection decisions incorporate ease of use, portability, and dependability issues as well as cost and performance issues. Even the selection of an information system project utilizing extensive quantitative cost/benefit methods such as break-even analysis, net present value, and rate of return should include qualitative factors such as availability of personnel, manageability and ease of maintenance in the decision process.

In spite of the necessity of considering qualitative elements, quantitative cost/benefit methods are most often proposed as tools for selecting among design alternatives. And while structured methodologies such as BSP [7] or Plexsys [14] recognize the need to consider qualitative attributes, they do not suggest the formal models for evaluation called for by Emery [4], instead leaving the task to the ingenuity of the decision maker.

Existing Approaches

Most selection problems involve multiple attributes, both qualitative and quantitative. Many methods exist to aid a decision maker in solving multiple attribute problems (see [6]) and have been adapted to handle qualitative attributes with varying degrees of success.

Bayesian analysis has been accepted by many developers. Bayesian analysis requires estimates of qualitative benefits and probability estimates for the measures in order to determine expected values. These estimates not only require the subjective estimates of the attribute being measured but multiply them by subjective probabilities. Subjective probability is difficult to assess and the decision maker is prone to violate the laws of probability [16]. The violations are not due to small random errors expected in subjective elicitation, but are errors of considerable magnitude. They result from perceptual and cognitive biases in subject's responses. For example, probability assignments are considerably larger if the relationship between events is perceived to be causal rather than diagnostic. This bias often forces the response beyond the limits dictated by the axioms of probability.

The use of expert systems to select hardware configurations from a set of user requirements is explored by Zeldin [21]. This approach removes the burden of choice from the decision maker and replaces the choice with a set of production rules. The expert system approach recognizes the lack of quantifiable attributes and allows for descriptive elements to be incorporated into the decision. However, a complete set of formal decision rules is required to be prepared for an inference device. In addition, a complete set of specifications are required to determine a valid hardware configuration. This process can be time consuming and beyond the resources of many firms, even though a sound design is the resultant output.

Another approach, applied to the selection of a development methodology by Neumann and Palvia [17], is to subjectively apply values to the qualitative attributes and to use a linearly weighted objective function to choose from among a set of alternatives. The method uses a solid rationale in applying a linear additive value function to the problem, but still relies on artificial measures for the descriptive attributes. An attribute list for each of the problems discussed may be found in the literature [2, 5, 17].

The application of a linear selection rule using subjective valuations can provide guidelines in a selection process. However, a decision resulting from such a procedure may be problematic. To apply the linear selection rule

requires that the problem be optimized using linear approximation. Estimation of the parameters of a linear function through subjective means could result in incorrect parameters. If the parameters of the linear decision rule are wrong, or if the decision maker's preferences are more correctly represented by a nonlinear function, then the optimization will lead to an inferior solution as demonstrated in [8].

Interactive articulation methods involve a decision maker interfacing with a computer. The decision maker is presented with a set of actual outcomes and asked to state a preference. Based upon that selection certain outcomes are eliminated using mathematical programming techniques. Interactive methods are receiving more attention as the power of computers increases and the decision maker becomes more active in the decision process. A review of the more common methods has been compiled by Hwang and Masud [6].

Most of the interactive methods require quantitative values for the criteria under consideration. An exception is a zero-one integer approach suggested by Klein and Hannan [11] and Bitran [1]. Using these methods, two values of a discrete, descriptive attribute could be represented by a single zero-one variable. Writing appropriate constraints and a sophisticated interface to the zero-one algorithm would permit the decision maker to solve a problem without resorting to subjective measures for the criteria. The limitations of the zero-one method includes the assumption of linearity, restriction to problems with certain outcomes, and complex solution techniques.

In order to overcome the difficulties mentioned, we turn attention to preference theory. In particular we consider prior articulation preference methods for multiple criteria decision making (MCDM). These methods include value function and utility function methods as described by Keeney and Raiffa [9]. These methods, however, ignore the existence of descriptive criteria.

Most preference methods can only incorporate descriptive attributes by rigorously obtaining a value or worth for each attribute outcome. This approach is common for methods that assume continuous values for the attributes. Thus, the major disadvantage of this

approach would be the need to subjectively value the criteria prior to their use in mathematical models. The final output from such an approach would, at best, be questionable.

In this paper, we propose using a variation on a method by Klein and Beck [12] that will require only the simplest of responses by the decision maker in the form of paired comparisons as used in other methodologies [19, 22]. The method is appropriate for problems with a discrete number of alternatives, with or without descriptive attributes, as in system design, hardware selection, or packaged software selection. The interactive algorithm utilizes a general, rather than a strictly linear value function.

The Formal Methodology

Value functions

Value functions are representations of incremental worth to a decision maker under known conditions. Value functions represent worth such that true preferences are linearly represented. For example, the first 10 megabytes of hard disk storage may be critical to the data operations of a company. The second 10 megabytes may be a convenience, but not a necessity. The third 10 megabytes may not add any worth to the overall product from the decision maker's viewpoint. Thus, the first 10 megabytes may represent 80% of the full value for hard disk storage while 19% of the value is on the second 10 megabytes. The remaining 1% of the total value is on any space above 20 megabytes. Figure 1 shows how this value function is represented. The value function for one attribute is called a single attribute value function (SAVF), $v(x)$, where x is the attribute level.

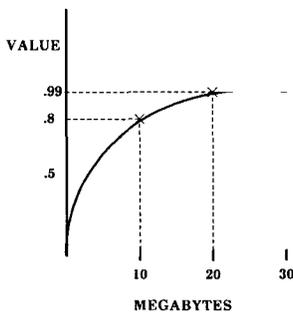


Figure 1. Single Attribute Value Function.

The importance of using value functions rather than actual attribute levels becomes evident using formal optimization aids. If the aids are used over actual attribute quantities, each unit will have equal value. Thus, if the first 10 megabytes of hard disk storage is worth \$800 to the decision maker, each successive unit will be worth \$800. But with value functions, the first 10 megabytes may be worth \$800 and the second only \$190. This ability to trade off worth rather than units is critical in many decisions.

In order to determine values when multiple attributes are present, the concept of multiple attribute value functions (MAVFs) is applied. MAVFs are mathematical combinations of the single attribute value functions. A common form is the additive MAVF:

$$V(X) = \sum_{i=1}^n w_i v_i(x_i)$$

where n = number of attributes,
 x_i = i th attribute,
 v_i = i th SAVF,
 w_i = a weight on the i th attribute,
 X = the entire attribute set, and
 V = the MAVF.

The additive MAVF is the weighted sum of the single attribute value functions. The multiple attribute value function provides a single measure that can be applied in an optimization routine to select the most valuable solution when the attributes are quantitative. The theory still holds, however, for qualitative attributes. The algorithm to be discussed will also work for qualitative attributes.

Independence conditions

In order to proceed, certain properties of the attributes must be established. The properties of interest are preferential independence and mutual preferential independence.

Preferential independence implies that a decision maker can structure his preferences for one attribute while holding the others constant and that the structure will hold for all levels of the remaining attributes. More formally:

"The set of attributes X is preferentially independent (PI) of the complementary set Y if and only if the conditional preference structure in the x space given y' does not depend

on y }. Or, X is PI of Y if, and only if, for some y'

$[(x', y') \text{ is preferred to } (x'', y')] \text{ implies that } [(x', y) \text{ is preferred to } (x'', y)] \text{ for all } y, x', x''.$ "
[9, p. 109].

This indicates that in order to have a valid value function $(v(x))$, the function must satisfy $v(x') > v(x'')$ regardless of y .

For example, if screen print color is preferentially independent of computer manufacturer and a consumer preferred a yellow print monochrome screen to a green print monochrome screen for an IBM PC, then the consumer would also prefer a yellow print monochrome screen to a green print monochrome screen for the TI Professional. If manufacturers were also preferentially independent of screen print color then the purchaser would be able to state:

"I prefer a yellow screen to a green screen, and an ALPHA 32 PC to an OMEGA 67 PC. Therefore, I want a yellow screen ALPHA 32 if my choices are a yellow screen ALPHA 32, a yellow screen OMEGA 67, a green screen ALPHA 32, and a green screen OMEGA 67."

Such reasoning is not contrary to application settings.

Preferential independence between only a subset of all attribute interactions is not sufficient for the methodology of this paper. A statement of this property in global terms is mutual preferential independence (MPI). The attributes x_1, x_2, \dots, x_n are mutually preferentially independent if every subset of these attributes is preferentially independent of its complementary set of evaluators. From an operational viewpoint, this means that preference statements made over any subset of attributes hold true for all constants of the remaining attributes. In the case of the hardware selection example, MPI implies that screen color and manufacturer would be independent of any other attributes that may be brought into the decision jointly or separately.

Theoretically, MPI means the existence of an additive multi-attribute value function. Although this form is restrictive, it is much more general than the linear restrictions placed on the value function by other methods [1, 5, 17,

22] since the single attribute value function can take many forms in their deviation from linearity. With the property of MPI in hand, preference statements for one attribute can eliminate alternatives from a solution space.

The algorithm

An algorithm is developed that utilizes preference theory in order to determine which alternatives are dominated. Dominated alternatives are eliminated preserving the integrity of value function theory. Dominance will be established by asking a series of questions requiring a statement of preference of one alternative to a second alternative over a subset of attributes.

The problem representation will be a tree similar to the branch and bound methods for integer programming [20], but will prune branches from an explicit tree through dominance considerations rather than expand the tree through branching. Since a discrete number of outcomes exist, convergence is guaranteed as each question will eliminate an alternative. This does not mean that convergence is rapid, merely that it will occur.

The method pictorially will work as follows. In Figure 2 we see a representation of a multiple objective problem with three objectives, each objective having multiple values. For simplicity, the example problem represents the selection of a graphics system when only system directives, screen positioning and system price are considered. All the outcomes are presented in Table 1. Also, in order to allow a complete explanation, price will not be assumed to have decreasing value as would be the case in a true application. The example will be solved in detail during and after the explanation of the algorithm.

In Figure 2, each final branch represents a different alternative from which a selection is required. The first attribute is of a descriptive nature, and a preference statement over two values of the first alternative is enough to eliminate not just one alternative, but every alternative with the inferior level of attribute 1 when attributes 2 and 3 are identical. Thus, if a menu-driven system is preferred to a command-driven system, then branch number 13 (alternative number 13) is inferior to branch number 5. Without asking further questions, it

Table 1. Matrix A for the Graphics Selection Problem

Product Number	System Directives	Screen Positioning	System Price (\$)
1	Menu-driven	Cursor keys	800
2*	Menu-driven	Cursor keys	500
3	Menu-driven	Cursor keys	700
4	Menu-driven	Mouse	800
5*	Menu-driven	Mouse	500
6	Menu-driven	Mouse	700
7	Menu-driven	Touch screen	700
8*	Menu-driven	Touch screen	500
9	Menu-driven	Touch screen	800
10	Command-driven	Touch screen	800
11	Command-driven	Touch screen	500
12*	Command-driven	Touch screen	600
13	Command-driven	Mouse	500
14*	Command-driven	Mouse	600

* Remain after 3 iterations

can be implied using established decision theory that branch 9 dominates branch 10 and branch 8 dominates branch 11. We progressively eliminate outcomes by asking other similar questions, the worst of which will be a paired comparison of two complete attribute vectors. If the situation arises where all attributes are included in a paired preference question, the decision maker may decide to

resort to other approaches, but many alternatives will already have been eliminated.

The following algorithm arrives at the best solution in a structured framework that will insure elimination of undesired outcomes:

Notation:

A — the decision table matrix

Attribute 1:
System Directives

Attribute 2:
Screen Positioning

Attribute 3:
Price (\$)

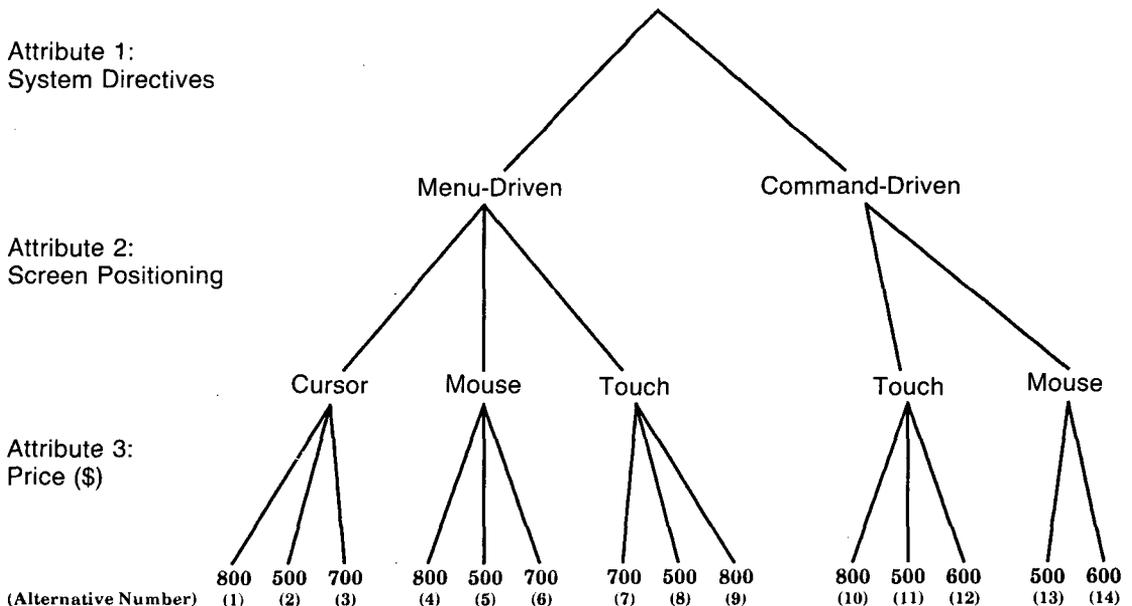


Figure 2. Alternative Tree

- a_{ij} —the i th attribute at the level in alternative j
- A^* —the attributes included in a preference question
- A^+ —the a_{ij} 's for a preference question from the first alternative
- A^- —the a_{ij} 's from the second alternative in a preference question
- m —the number of alternatives
- n —the number of attributes

Steps of the algorithm:

STEP 1. Initialize. Generate the matrix A . Include all feasible alternatives with their attribute values. Matrix A is essentially a list of all the alternatives in the decision process as in Table 1 for the example. As the number of alternatives increases, the size of A will increase.

STEP 2. Termination Test. If $M=1$ then stop. When the number of alternatives remaining in the decision is one, stop. It may be desirable to stop the algorithm when three or four alternatives remain. A more formal economic study could then be conducted on the candidates remaining after the method is used to quickly eliminate all but a small subject of alternatives.

STEP 3. Select Attributes to Appear in the Paired Comparison Question. Select A^* (the attributes to include in a preference question). A^* will be a 1 to n attribute. The attributes included in A^* must uniquely separate at least 2 alternatives in such a fashion that the remaining attributes in the alternatives of selection are constant. Let those alternatives be denoted X and Y . For the example problem in Table 1, system directives distinguish products 9 and 10 since their remaining attributes are identical. Thus A^* is system directives, X is product 9, and Y is product 10 in the first page of the algorithm.

STEP 4. Question the Decision Maker. Determine A^+ as the collection of a_{ij} for every i an element of A^* , for j equal to the i th attribute in alternative X . Determine A^- as the collection of a_{ij} for every i an element of A^* , for j equal to the i th attribute in alternative Y . Ask the DM for his preference of A^+ and A^- . In the first pass of the example, A^+ is set to be the attribute "system directives" at the level "menu-driven" as taken from product 9. A^- is set to be the attribute "system directives" at

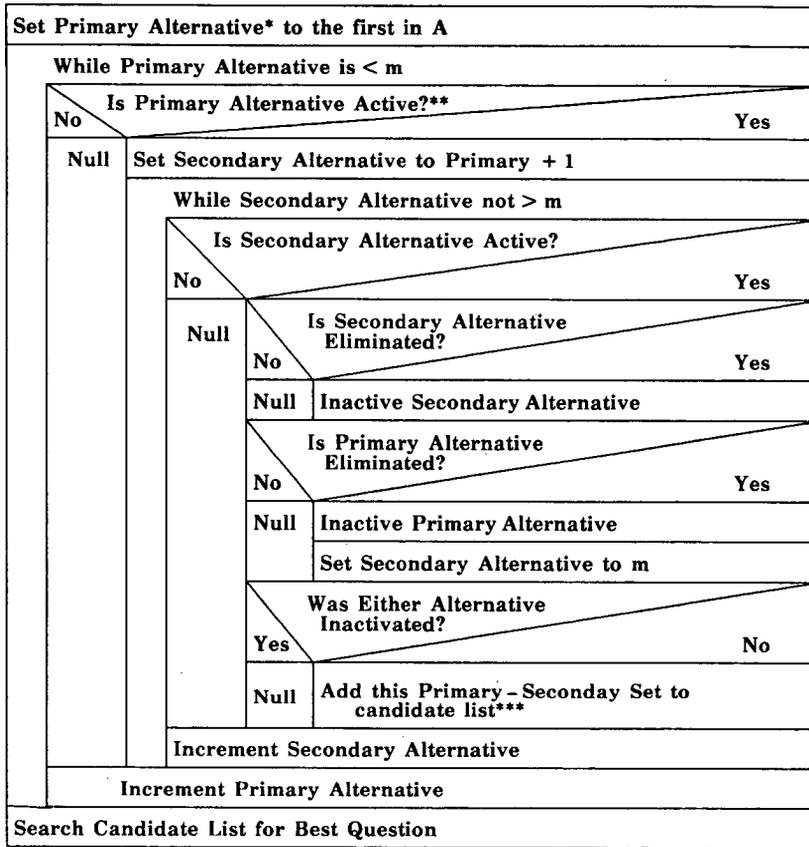
the level "command-driven" as taken from product 10. The decision maker is asked for a preference between A^+ and A^- , or between menu-driven and command-driven systems.

Even though the attribute set is divided into smaller sets, the decision maker must keep the problem setting in mind. The question is not "Do you prefer menu-driven systems or command-driven systems?" but "For your graphics applications are menu-driven systems more functional than command-driven systems?"

STEP 5. Eliminate Dominated Alternatives. For every alternative pair in A where A^+ and A^- distinguish the two alternatives, eliminate the least preferred from A . Return to step 2. For the example's first pass, all products distinguished *only* by their system directive properties are eliminated from Matrix A . Continuing with Table 1, if the decision maker prefers menu-driven systems to command-driven systems, then products 10, 11, and 13 are known to be dominated by products 9, 8, and 5 respectively.

In order to find every alternative eliminated, each pair of alternatives must be examined. This involves $(m-1)(m)/2$ vector comparisons. For a small matrix this presents no problem. As A increases in size, however, the system response time could become excessive. It may be desirable to terminate the search after a certain time in order to respond to the decision maker in short order. This would place a cap on response time at the expense of asking the DM an extra preference question. The proper way to handle the response times will depend on the attitude and patience of the DM. For a small Matrix A , however, the issue would not arise. Logic for the elimination process of step 5 is presented in the Nassi-Schneiderman chart in Figure 3.

Remainder of Example. In the second pass, step 2 would fail the stop test. Step 3 of the second iteration could find that A^* may be composed of System Price. Price separates products 1 and 2. Step 4 would then define A^+ as \$800 and A^- as \$500. If the decision maker prefers \$500, then Step 5 will eliminate products 1, 4, and 9. Using the same process for the third iteration, a statement by the decision maker that \$500 is preferred to \$700 would eliminate products 3, 6, and 7 from consideration, leaving us with only the options



- * Primary and Secondary refers to loop control and not any alternative preferences.
- ** Active means the alternative has not been previously eliminated from A.
- *** The candidate list includes alternative pairs that could provide questions in Step 3.

Figure 3. Alternative Elimination Logic

shown by the alternatives in Table 1 marked by an asterisk (*).

The next attribute that should be used is screen positioning. Screen positioning separates products 5 and 8 with the values "Mouse" and "Touch Screen." Thus, if the decision maker prefers screen positioning using touch screen over a mouse, the algorithm will eliminate alternatives 5 and 14. If in the fifth iteration similar logic is used and the decision maker prefers a mouse over cursor keys, then product 2 is out of the running.

Preparation for the sixth iteration reveals that there is no single attribute that separates the

remaining outcomes. However, two attributes separate the remaining outcomes. Thus, A* must be composed of two attributes in Step 3, those being system directives and price. Step 4 defines A⁺ as "menu-driven, \$500" and A⁻ as "command-driven, \$600." This preference statement by the decision maker will eliminate all but one alternative and stop upon returning to Step 2.

What is important to note is that the most simple preference questions were first asked in order to reduce problem size. More complex questions were asked later, but never containing all the attributes and never of great quantity. This control of question complexity

is conducted in Step 3 of the algorithm during the selection of A*.

A* may be selected in several fashions. One method is to continually examine the two alternatives at the top of Matrix A. However, this will continually increase the number of attributes that separate the top two alternatives and thus increase the difficulty of the questions to which the decision maker must respond. A better approach would be to search Matrix A for a pair of alternatives that minimize the number of attributes in A*. This would provide the fewest attributes in each question and yield the preference questions easiest to answer. The alternative elimination of Step 5 requires the use of such a search and can conduct a look ahead for Step 3 of the next iteration.

Another implementation issue is when a large number of outcomes exist within the initial alternative matrix A. Reduction measures can be taken. If a measurable attribute exists, it may prove swifter to first measure the single attribute value function (SAVF). Rapid procedures for this are available [10, 13]. The effect of this action is to place a prior preference on an attribute, such as the attribute price. The SAVF can then be applied to generate automatic responses to the pairwise preference questions involving the attribute with the known SAVF. If this process can be accomplished for more than one attribute then eliminations may be made on automatic responses to the pairwise preference questions over several attributes and the number of questions asked of the decision maker will decrease.

Conclusions

A method is presented to aid in multiple attribute decisions involving descriptive, qualitative attributes. The basic theory is outlined prior to describing the method in a step by step fashion. An example of software selection is provided to highlight each step.

The type of problems that may be solved with the technique described are numerous. Software selection problems involve attributes of price, directives, and accuracy that may not be quantifiable. Hardware selection includes dependability issues that cannot have a number attached. Many selection decisions at

various points in the system development process fall into this class of problems and the decision process can be assisted with a formal aid.

The method is a step by step approach that requires interaction with a decision maker. At certain steps, the decision maker is asked to state a simple preference of one alternative from a set of two alternatives. This preference is usually made over only a subset of the total number of attributes. This feature keeps the questions asked of the decision maker as simple as possible, yet provides the information required to eliminate multiple selections based upon a body of mathematical theory.

In addition to keeping the interactions simple, they are also deterministic. Thus, the method is easy to install as a computerized, interactive decision aid that can be added to an organization's set of automated design and decision tools.

The algorithm is not appropriate for situations involving risk or where continuous variables exist in the problem. Research is under way to investigate extensions in these areas. The method may also not be effective for decision situations with large amounts of economic data and only a few alternatives, since to effect proper responses from the decision maker would require a study involving more standard cost/benefit procedures. But still, many selection problems can be aided with proper application of the proposed method.

References

1. Bitran, G.R. "Theory and Algorithms for Linear Multiple Objective Programs with Zero-One Variables," *Mathematical Programming*, Volume 17, Number 3, November 1979, pp. 362-390.
2. Boehm, B. *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1981.
3. Couger, J. D. "The Benefit Side of Cost Benefit Analysis, Part II," *Data Processing Management Portfolio 1-01-08*, Auerbach Publishers, Inc., Pennsauken, New Jersey, 1975, pp. 1-13.
4. Emery, J. "Cost/Benefit Analysis of Information Systems," in *Advanced System Development/Feasibility Techniques*, J. D. Couger, M. A. Colter, R. W. Knapp

- (eds.), John Wiley & Sons, New York, New York, 1982, pp. 459-488.
5. Gore, M. and Stubbe, J. *Elements of Systems Analysis*, William C. Brown Co., Dubuque, Iowa, 1983.
 6. Hwang, C. L. and Masud, A. S. M. *Multiple Objective Decision Making-Methods and Applications*, Springer-Verlag's Lecture Notes in Economics and Mathematical Systems #164, Springer-Verlag, Berlin, Germany, 1979.
 7. IBM. "Business Systems Planning," IBM, Document GE20-0527 White Plains, New York, 1979.
 8. Keefer, A. L. and Pollock, S. M. "Approximations and Sensitivity in Multiobjective Resource Allocation," *Operations Research*, Volume 28, Number 1, January-February 1980, pp. 114-128.
 9. Keeney, R. L. and Raiffa, H. *Decisions with Multiple Objectives*, Wiley, New York, New York, 1976.
 10. Kirkwood, C. W. and Sarin, R. K. "Preference Conditions for Multi-Attribute Value Functions," *Operations Research*, Volume 28, Number 1, January-February 1980, pp. 225-232.
 11. Klein, D. and Hannan, E. "A Branch and Bound Algorithm for the Multiple Objective Zero-One Linear Programming Problem," *European Journal of Operations Research*, Volume 9, Number 3, June 1982, pp. 378-385.
 12. Klein, G. and Beck, P. "A Formal Multiple Criteria Decision Aid For Descriptive Attribute Problems," MIS Working Paper, Southern Methodist University, Dallas, Texas, 1985.
 13. Klein, G., Moskowitz, H., Mahesh, S. and Ravindran, A. "Assessment of Multi-Attributed Measurable Value and Utility Functions Via Mathematical Programming," *Decision Sciences*, Volume 16, Number 3, July 1985, pp. 309-324.
 14. Konsynski, B., Kottemann, J., Nunamaker, J. and Stott, J. "PLEXSYS-84: An Integrated Development Environment for Information Systems," *Journal of Management Information Systems*, Volume 1, Number 3, Winter 1984-1985, pp. 64-104.
 15. Lucas, H. *The Analysis, Design, and Implementation of Information Systems*, McGraw-Hill, New York, New York, 1985.
 16. Moskowitz, H. and Sarin, R. "Improving the Consistency of Conditional Probability Assessments For Forecasting," *Management Science*, Volume 29, Number 6, June 1983, pp. 735-749.
 17. Naumann, J. and Palvia, S. "A Selection Model for System Development Tools," *MIS Quarterly*, Volume 6, Number 1, March 1982, pp. 39-48.
 18. Powers, M., Adams, D. and Mills, H. *Computer Information Systems Development: Analysis and Design*, South-Western Publishing Co., Cincinnati, OH, 1984.
 19. Sadogapan, S. and Ravindran, A. "Interactive Solution of Bi-criteria Mathematical Programs," *Naval Research Logistics Quarterly*, Volume 29, Number 3, September 1982, pp. 443-459.
 20. Salkin, H. *Integer Programming*, Addison-Wesley, Reading, Massachusetts, 1975.
 21. Zeldin, P. *An Expert Model Based on User/Process Profiles for Computer Systems Evaluation*, Unpublished Ph.D. Dissertation, University of Arizona, Tucson, Arizona, 1984.
 22. Zionts, S. and Wallenius, J. "An Interactive Programming Method for Solving the Multiple Criteria Problem," *Management Science*, Volume 22, Number 6, June 1976, pp. 652-663.

About the Authors

Gary Klein is an Assistant Professor of Management Information Sciences in the Edwin L. Cox School of Business at Southern Methodist University. He received his Ph.D. in management science from Purdue University. His major research focus is model management techniques and applications. A secondary area of interest is the development of formal models and optimization procedures for information systems design.

Philip O. Beck is a Senior Operations Research Analyst at American Airlines. He received his Ph.D. in management science at the University of Texas, Austin. His major research interests include applications of management science to inventory management, scheduling, statistics, and MIS.