# A STUDY OF COMPUTER-BASED TECHNIQUES FOR MULTI-DIMENSIONAL EVALUATION IN URBAN PLANNING

by

THOMAS E. MARTIN
B. Arch., University of Toronto
(1967)
M. Arch. U.D., Harvard University
(1971)

Submitted in
Partial Fulfillment
of the Requirements for the
Degree of Master of City Planning
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September, 1971

Signature of Author..........................................
Department of Urban Studies and Planning
September 24, 1971

Certified by..........................................
Thesis Supervisor

Accepted by..........................................
Chairman, Departmental Committee on Graduate Students

C. 2

## ABSTRACT

A STUDY OF COMPUTER-BASED TECHNIQUES FOR MULTI-DIMENSIONAL EVALUATION IN URBAN PLANNING

Thomas Edmond Martin
Submitted to the Department of Urban Studies and Planning on September 24, 1971, in partial fulfillment of the requirements for the degree of Master of City Planning.

The thesis is directed towards the development of a computer-assisted capability for the evaluation of planning projects with multi-dimensional consequences. Evaluation models and routines are implemented in DISCOURSE, an on-line computer language oriented towards spatially disaggregated environmental design problems.

A variety of issues in the evaluation of complex problems are introduced: the role of evaluation in the planning process; relations between design descriptors and evaluators; the multi-dimensionality and hierarchicization of goals; preferences for value, risk, and time; and the representation of predicted consequences in an impact matrix. This discussion forms the basis for a taxonomy of multiple objective preference models which range from simple ordering of consequences to complex multi-dimensional utility theory. Preferences for certain consequences with no tradeoffs among evaluators, tradeoff analysis under certainty, and multi-dimensional preferences for risky consequences, are outlined.

The next section develops hierarchical systems models in more detail, describing three functional forms: decision complexity, description and organization. A distinction is made between hierarchical goal models, structured in terms of decision complexity, and hierarchical planning models differentiated by levels of description or abstraction. A number of hierarchical goal models are described and related to multi-dimensional preference structures; Manheim's Hierarchical Structure is discussed as an example of hierarchical planning models; and from this, desirable characteristics of a multi-dimensional, hierarchically structured evaluation system are developed.

A computer-aided evaluation system is presented, with capabilities in three areas:

(a) "User Operations", a set of flexible, independent routines for manipulating a design impact matrix;
(b) "Static Evaluation", a terminal assessment procedure, with relative value, certainty, and risky, preference models;
(c) "Dynamic Evaluation", a hierarchically structured planning model, operating on both goal and design structures.

All three groups of programs accept design alternatives which have been generated at hierarchical levels of generality, but this is a necessary requirement only for the Dynamic Evaluation model. However, if design alternatives have been so structured, then a corresponding goal structure must also be input. M.I.T.'s North West Area Project is used as an illustrative experiment for the testing of the component evaluation routines.

Extensions of this work to include considerations of social welfare and social choice, user participation and gaming, cost-benefit analysis and preferences for time, and incorporation of evaluation techniques within a larger and more comprehensive evaluation strategy, are also suggested.

| | |
|---|---|
| Thesis Supervisor: | William L. Porter |
| Title: | Associate Professor of Urban Design<br>Department of Urban Studies and Planning |

ACKNOWLEDGEMENTS

Professors Ralph Keeney and Aaron Fleisher ably provided critical comments and corrections for much of the material; special gratitude is due to Professor Marvin L. Manheim for his inspiration and comments on the thesis, much of which traverses areas he has thoroughly explored.

Wren McMains and Tim Lundeen provided many hours of programming assistance in DISCOURSE and PL/1, and corrected numerous personal and system idiosyncracies. Art Lidsky of the M.I.T. Planning Office graciously allowed many interruptions of his time to answer questions and provide information and data about the North West Area Project. Brian Kemerer aided in programming prediction routines for the North West Area Project, which serves as an illustrative empirical application for the ideas in the thesis.

A substantial debt of thanks is owed to Central Mortgage and Housing Corporation, Ottawa; which, while not directly supporting the work herein, has materially assisted my graduate work in urban studies over the past three years, through its program of Graduate Fellowships in Urban and Regional Affairs. Computer time for the project was provided through the Ford Foundation - Urban Grant to the Department of Urban Studies and Planning.

Professor William L. Porter has provided assistance, stimulation, advice, and encouragement in his role as advisor and guru.

Barbara Barnes typed quickly and faultlessly in the preparation of the thesis.

My wife Kristine, as always.

TABLE OF CONTENTS

## 1. INTRODUCTION

The solution to a problem pre-supposes several conditions:

(a) a decision language or formal system in which the immediate problem may be disconnected from its larger context and stated unambiguously; i.e. the "problem representation";

(b) a set of computational procedures within a "plan", which operate on an initial problem statement A, to transform it into a succeeding state A': i.e.

$$A \Longrightarrow A'$$

These are commonly termed "search" procedures;

(c) a set of criteria for determining when a given problem transformation is satisfactory to the decision-maker. This testing of design alternatives for their suitability is termed "evaluation".

We can concentrate on evaluation as an issue in urban problem-solving because most environmental problems are "ill-defined", i.e. with no systematic means of deciding when a proposed solution is acceptable. To illustrate, we adopt Reitman's (1) notation for problem analysis:

---

(1) W. R. Reitman; "Heuristic decision procedures, open constraints, and the structure of ill-defined problems", in M. W. Shelley, III & G. L. Bryan, eds.; Human Judgments and Optimality, (New York, N. Y., John Wiley & Sons, Inc., 1964), Ch. 15, pp. 282 - 315.

A = an initial problem state

B = a transformed problem state

$\Rightarrow$ = a process, program, or sequence of operations for transforming A into B

In classical, unconstrained optimization (as in calculus), where A, B, and $\Longrightarrow$ are all well defined, evaluation is implicit in the conditions for solution. In well-defined optimization problems under constraints, where the solution procedure is an iterative algorithm, separate search and evaluation components may be distinguished within the same iteration: "search" generates a transformation of the present alternative; "evaluation" tests for optimality. In ill-defined problems, where any, or all of: A, the initial state; B, the transformed state; or $\Longrightarrow$, the set of available operations; may be vaguely defined (if at all), we must concentrate on elaborating the decision-maker's choice criteria in order to determine when a given solution is satisfactory. In this latter case, separate search and evaluation phases may be distinguished in the planning process.

Within search routines, a number of implicit or internal tests may be embedded. However, we will discuss only "external" tests; i.e. the evaluation of alternatives with respect to explicit goal statements. Three evaluation models are developed, each corresponding to different roles that evaluation may take on in the planning process. Figure No. 1.1 illustrates Reitman's concept of the problem-solving process as a series of successive transformations of problem states:

$$A \Longrightarrow A' \Longrightarrow A'' \Longrightarrow \ldots\ldots\ldots A^{i} \Longrightarrow \ldots\ldots\ldots B$$

initial state

terminal state

Figure No. 1.1

Each node in the path is a problem vector which satisfies the constraints implied by the attributes of the vector preceding it in the chain. Conversely, each transformation defines a set of constraints that must be met by subsequent transforms if they are to lead to a solution of that problem. (2)

Evaluation procedures may be applied after any particular transformed state $A^{i}$, in order to assess some aspect of the process. The purpose of such evaluation may be to determine what transformation to undertake next in the process. Such comparative procedures, applied "in process", we call "user operations". If B, the terminal state, consists of a number of alternative transformed states, evaluation assesses which of the alternatives is most satisfactory to the decision-maker. We call this terminal assessment procedure "static evaluation". Finally, evaluation procedures may be incorporated within a larger "meta-procedure" which guides the planning relations among sequences of transformations; this is termed "dynamic evaluation".

(2) Ibid.; p. 305.

Manheim has defined evaluation as:

> "...the process of arraying and aggregating the consequences of an action to facilitate decision-making."(3)

The basic input component to an evaluation procedure is a set of predicted consequences, usually arrayed in an impact matrix. Design consequences are derived from descriptor attributes of the design, through transformation by a set of prediction operators: (cf. Figure No. 1.2). Prediction is intended to anticipate the consequences which would result if the design were to be actually implemented.

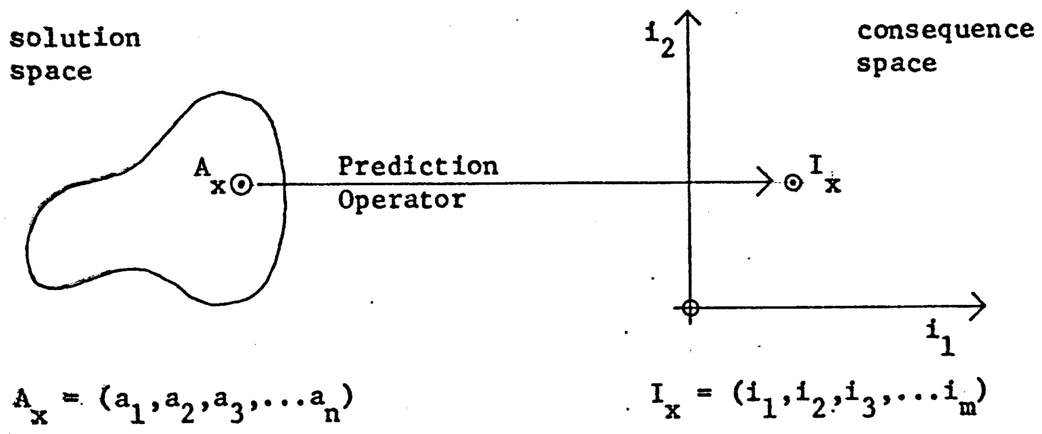


Figure No. 1.2

Each design alternative, $A_x = (a_1,a_2,a_3,...a_n)$ is associated with a unique point in n-dimensional attribute space, and is mapped onto a unique point $I_x = (i_1,i_2,i_3,...i_n)$ in m-dimensional consequence

---

(3) M. L. Manheim, et. al.; The Impacts of Highways upon Environmental Values, (Cambridge, Mass., M.I.T. Urban Systems Laboratory, Report No. USL-69-1, March, 1969), p. 37.

space (under certainty), or a unique set of m probability distributions over consequences (under risk). Each consequence that is associated with a facet of the decision-maker's preference structure, we term an "evaluator". (In our efficient and parsimonious view of the planning process, the decision-maker predicts only those consequences which are relevant to evaluation.) An evaluator is transformed consequence; at the very least, a consequence ordered to reflect direction of preference. However, the distinction between design "attributes" and "evaluators" is not always clear because of the phenomenon of "constraint proliferation" in the planning process, as suggested by Reitman:

> "all attributes of any object or process introduced into the problem may serve as constraints on the solution....As problem solving proceeds, the progressively more differentiated problem components themselves become increasingly more important as a source of constraints." (4)

A large number of design attributes are left "open" (i.e. with one or more parameters left unspecified) at the beginning of the process. The assignment of a set of values to design attributes reduces the size of the search space, within which, successively more detailed alternatives are developed. This use of attribute values as temporary constraints has primarily local implications for guiding search; they become the criteria for the internal tests mentioned above. In contrast, "evaluators" are the global criteria by which the external tests operate, to assess alternatives with

(4) W. R. Reitman; op. cit., p. 297.

respect to explicit goal statements.

The result of "globally" evaluating a set of alternatives with respect to a set of goal variables, is a one-dimensional ordinal ranking of this set; thus, for multi-dimensional problems, with more than one consequence, evaluation necessarily entails a condensation or reduction in this dimensionality. Depending on its nature, the condensation or aggregation of consequences must be done by different means, so as to reduce possible arbitrariness introduced by the loss of information content. Spatial statistical distributions may be used to summarize over spatially-disaggregated consequences. (5) Political bargaining and logrolling processes may be required for aggregation over a number of impacted actors or community groups. In one-dimensional utility theory, aggregation over probabilistically distributed consequences is done through the probability calculus. In this paper, we focus on the aggregation of consequences over a number of goal dimensions, through multi-dimensional value and utility theory. Condensation of monetary consequences distributed over time may be done by standard discounting formulae, though multi-dimensional utility theory is applicable here also.

Evaluation also involves transformations of consequence space, the extent of which, depends on how strongly the decision-maker has elaborated his preferences. These "preferences", or statements

(5) D. S. Neft; Statistical Analysis for Areal Distributions, (Philadelphia, Pa., Regional Science Institute, Monograph series, No. 2, 1966).

about desirable states of the world, may take on three different aspects:

(a) value (the numerical level of a consequence);

(b) risk (probability distributions of consequence values);

(c) time (when the consequence occurs).

Preferences for time are not discussed here, but are well developed in literature on cost-benefit analysis (6). A taxonomy of preference models for value and risk, is developed in Section 2.

The result of the transformation for each consequence $i_j$, is an associated worth index, $v_j$:

$$\underset{\text{consequence } j}{(i_j)} \xrightarrow{v_j \;=\; u_j(i_j)} \underset{\text{value } j}{(v_j)}$$

The interpretation given to this worth index depends on two kinds of measurement:

(1) the accuracy of the value assigned to the predicted consequence (a function of the prediction model and its associated measurement scale);

(2) the discrimination and scaling of the decision-maker's preferences with respect to the predicted consequence.

---

(6) for example: A. Maass & M. Hufschmidt; Design of Water Resource Systems, (Cambridge, Mass., Harvard University Press, 1962).

The concept of "measurability" is crucial to evaluation, since the real purpose of measuring is to be able to predict certain events (such as choices). If the measures of events are ambiguous, they must either be accepted as such (i.e. as a property of the events), or the ambiguity must be removed, since "decidability" and ambiguity cannot co-exist in the same problematic context.

The relationship between measurement of consequences and of preferences will not be discussed here, nor will theoretical bases for measurement and scaling. (7) Comparability of all alternatives with respect to the same evaluators is required, since evaluation introduces a consistent form of comparability among alternatives. In some contexts, ordinal measures may be sufficient for decidability, however ordinal scaling is very limited, relative to a specific set of alternative outcomes, and ambiguous outide this set. Most complex evaluation situations require interval or higher measures, both of consequences and of preferences (such as the von Neumann-Morgenstern interval utility scale (8)). Fishburn (9) develops an extensive set of theorems for "ordered metric" measures in the domain between ordinal and interval scaling, but again,

---

(7) for example: C. H. Coombs, H. Raiffa, R. M. Thrall; "Some Views on Mathematical Models and Measurement Theory", in Thrall, Coombs, & Davis, eds.; Decision Processes, (New York, N. Y., John Wiley & Sons, Inc., 1954), pp. 19 - 37; or W. S. Torgerson; Theory and Methods of Scaling, (New York, Wiley, 1958).

(8) J. von Neumann & O. Morgenstern; Theory of Games and Economic Behaviour, (Princeton, N. J., Princeton University Press, 1947).

(9) P. C. Fishburn; Decision and Value Theory, (New York, N. Y., John Wiley & Sons, Inc., 1964).

such measures are relative to a specific set of outcomes. Decidability, i.e. the unambiguous selection of a preferred alternative, depends on the lowest level of measurement in a problem context. Though not wishing to sidestep a complex issue, for the purposes of this paper, we assume that consequences and preferences are measurable to the level required for unambiguous choice (i.e. usually interval scaling). This assumption also implies acceptance of a number of normative axioms such as transitivity, closure, continuity, monotonicity, etc., (10) underlying measurement models; principles which may be difficult to accept in a complex empirical situation. Thus, we temporaily disregard the subtle interplay between descriptive and normative decision criteria.

Evaluation procedures should aim for economy in information acquisition and processing, in the sense that dominance should be established with the use of the lowest scale of measurement consistent with unambiguous choice, since this makes the fewest demands on the decision-maker. Higher measurement scales should be invoked only when necessary to resolve these ambiguities. However, this is almost entirely dependent on the structure of the problem: when choice is clear-cut, the use of an "evaluation method" is trivial; when it is ambiguous, the selection of method depends on the nature and extent of this ambiguity.

(10) for example: Coombs, Raiffa, & Thrall; op. cit.

The concept of hierarchical levels of evaluation is related to differentiation of measurement scales. Hierarchization is synonymous with multi-dimensionality, since general goals usually have to be disaggregated into multiple operational objectives when the former are not measurable directly. If they are measurable, upper level goals are more likely to be assessed on ordinal or nominal scales, whereas lower level objectives are likely to be measured on ordered metric or higher scales.

Analogously, alternatives may be developed at several hierarchical levels of detail, at different stages throughout the planning process. Whether or not a hierarchical planning process will be used to generate alternatives, depends on the degree of interdependence among different sub-problem components. If (as is rarely the case) alternatives can be generated from the simple aggregation of solutions to a number of subproblems (e.g. as in linear programming), then design may proceed directly to the solution of these components. On the other hand, if there is a good deal of interdependence among sub-problem solutions, which precludes their simple aggregation, both global and local aspects of the problem must be considered together, throughout the process. In this latter case, a hierarchically structured approach to the generation of alternatives may be feasible. How accurately these intermediate alternatives can be evaluated, depends on the precision of the prediction models, which in turn, relates to the number of design attributes and their level of measurement.

Preferences can be more detailed for consequences which can be measured accurately. Possible relationships between hierarchical evaluation and the hierarchical generation of design alternatives, are discussed further in Section 3.

To conclude, we consider again, the role of evaluation in the planning process. Above, we outlined three different roles for evaluation techniques:

(1) "user operations";

(2) "static", terminal evaluation;

(3) "dynamic" evaluation.

More fundamentally, these models also serve several more detailed functions:

(1) Representation of design consequences and actor preferences. Consequences are displayed in an "impact matrix" which serves as a basis for operations by various evaluation techniques. In its simplest form, an impact matrix has the following elements: (cf. Figure No. 1.3)

Alternatives

$$\begin{array}{ll|cccccc|}
 & & A_1 & A_2 & \cdots & A_j & \cdots & A_n \\
\text{Evaluators} & E_1 & i_{11} & i_{12} & \cdots & i_{1j} & \cdots & i_{1n} \\
 & E_2 & i_{21} & i_{22} & & \vdots & & \vdots \\
 & \vdots & \vdots & & & \vdots & & \vdots \\
 & E_i & i_{i1} & \cdots & \cdots & i_{ij} & & \vdots \\
 & \vdots & \vdots & & & & & \vdots \\
 & E_m & i_{m1} & \cdots & \cdots & i_{mj} & \cdots & i_{mn}
\end{array}$$

Figure No. 1.3

This basic format may be elaborated to include uncertainty (by associating probabilities $p_{ij}$ with every impact $i_{ij}$), differentiation of impact types (e.g. costs, quantitative effects, political effects, etc.), differentiation of actors (as a preliminary to a community bargaining process): (cf. Figure No. 1.4).

$$\begin{array}{llcl}
 & & \text{Alternatives} & \text{Actors} \\
 & & A_1 \cdots\cdots A_j \cdots\cdots A_n & \\
\text{Evaluators} & \begin{matrix} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_k \end{matrix} & \left[\begin{matrix} i_{11} & \cdots i_{1j} \cdots & i_{1n} \\ \vdots & & \vdots \\ \vdots & i_{ij} & \vdots \\ \vdots & & \vdots \\ i_{k1} & \cdots\cdots & i_{kn} \end{matrix}\right] & \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \\ a \end{matrix}}\right\} A \\
 & \begin{matrix} E_1 \\ \vdots \\ E_m \\ \vdots \\ E_p \end{matrix} & \left[\begin{matrix} i_{11} & \cdots i_{1j} \cdots & i_{1n} \\ \vdots & & \vdots \\ \vdots & i_{mj} & \vdots \\ \vdots & & \vdots \\ i_{p1} & \cdots\cdots & i_{pn} \end{matrix}\right] & \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \\ a \end{matrix}}\right\} B \\
 & \begin{matrix} E_r \\ \vdots \\ E_s \\ \vdots \\ E_t \end{matrix} & \left[\begin{matrix} i_{r1} & \cdots i_{rj} \cdots & i_{rn} \\ \vdots & & \vdots \\ \vdots & i_{sj} & \vdots \\ \vdots & & \vdots \\ i_{t1} & \cdots\cdots & i_{tn} \end{matrix}\right] & \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \\ a \end{matrix}}\right\} C
\end{array}$$

Figure No. 1.4

differentiation of actor groups (e.g. principal actors, secondary actors, special interests, etc.), tradeoffs among evaluators, (cf. Figure No. 1.5), etc.

$$\begin{array}{llcc} & & \text{Trade-off Ratio} & \text{Alternatives} \\ & & & \begin{matrix} A_1 & \cdots\cdots & A_j & \cdots\cdots & A_n \end{matrix} \\ \text{Evaluators} & \begin{matrix} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{matrix} & \begin{bmatrix} w_1 \\ \vdots \\ w_i \\ \vdots \\ w_m \end{bmatrix} & \begin{bmatrix} i_{11} & \cdots & i_{1j} & \cdots & i_{1n} \\ \vdots & & \vdots & & \vdots \\ i_{i1} & \cdots & i_{ij} & & \vdots \\ \vdots & & & & \vdots \\ i_{m1} & \cdots & \cdots & \cdots & i_{mn} \end{bmatrix} \end{array}$$

Figure No. 1.5

(2) <u>Comparison</u> of a set of impact matrix elements with respect to some other differentiated dimension (e.g. alternatives with respect to evaluators, actors, or preference functions; evaluators with respect to alternatives, actors, etc.)

(3) <u>Guidance</u> of the planning effort: display of crucial decision issues (e.g. points of agreement or disagreement among actors, similarities or dissimilarities between alternatives, unsatisfied goal variables, etc.); selection of design attributes for incremental improvement, bases for negotiation, etc.; derivation of tradeoffs or rough preference information from the decision-maker; indication of decision nodes for information acquisition and experimentation, etc.

(4) <u>Computation</u> and aggregation of worth indices and rankings of alternatives with respect to evaluators and actors, summary statistics, tradeoffs, analysis costs, etc.

(5) Self-organization: derivation of new preference structures, changing the dimensions of evaluation or search space; guidance of the search effort towards sub-optimality; optimal control of analysis resources.

The last point is suggested as a direction for further research, but is not within the scope of the paper. Evaluation techniques and strategies may serve some or all of the above functions, with variations from stage to stage in the planning process; from problem context to context; and from model to model. The proposed evaluation models provide for a range of responses to these functions.

## 2. MULTI-DIMENSIONAL PREFERENCE MODELS

A number of assumptions are introduced to simplify and shorten the discussion to follow. We have already mentioned the requirements of strict comparability of alternatives, and measurability of consequences. Others include:

(1) evaluation and search spaces are fixed for the duration of the planning process;

(2) the "social choice" problem (i.e. construction of a fair and acceptable ranking over alternatives for a large number of actors), and the "social welfare" problem (i.e. the equitable distribution of the costs and benefits of alternatives to all impacted actors) are not considered. Therefore, we assume a unitary decision-maker, or rather, the construction of a goal fabric which integrates the interests of all significant actors.

(3) goals can be disaggregated or decomposed to the detail required for measurable performance indices; preference information can be derived and assessed meaningfully.

(4) alternatives are assessed in terms of only two dimensions: consequences "x" and "y"; these consequences can be measured on a continuous (interval) scale, though a decision-maker's preferences for them may vary in precision.

Preferences for multi-dimensional consequences take on two aspects:

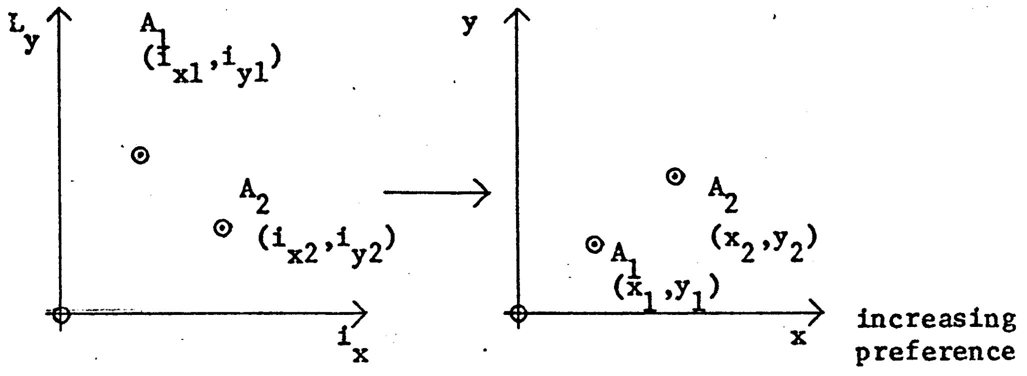
(1) preferences for value, risk, and time, of individual consequences, as developed in unidimensional utility theory;

(2) preferences or tradeoffs among types of consequences or dimensions.

In this section, these aspects are arrayed roughly in order of increasing demands made on the decision-maker's preference structure; that is, in terms of increasing transformations of consequence space. For certain outcomes with no risk or time dimensions, models range from very rough preferences with no implied tradeoffs among evaluators (e.g. ordering of consequences) to complex indifference curve analysis with detailed value and tradeoff preferences. The rudiments of multi-dimensional utility theory are developed for the added dimension of risk. Preferences for time are not discussed; practically, most theory in this area concentrates on single evaluators (usually monetary), introducing multiple time periods as the extra dimensions. Consideration of both multiple goals and multiple time periods quickly builds up dimensionality to unmanageable proportions.

We consider first, preference models for certain consequences, with risk and time considerations suppressed, and no implied tradeoffs among evaluators. The simplest form is the ordering of consequence space, i.e. the specification of directions of preferences: (cf. Figure No. 2.1)

## 2.1 Ordering of Consequences



Consequence Space

Ordered Consequence (Evaluation) Space

Figure No. 2.1

Dominance of alternative $A_2$ over $A_1$ is defined if $x_2 > x_1$ and $y_2 \geq y_1$; or $x_2 \geq x_1$ and $y_2 > y_1$; and vice versa for dominance of $A_1$ over $A_2$. Dominated alternatives are eliminated from further consideration. If this comparison is repeated over a large number of alternatives, a set of alternatives in which no alternative completely dominates any other, results. This set is called:

## 2.2 Pareto-Efficient Frontier

Figure No. 2.2 illustrates the Pareto frontier for a finite number of alternatives:

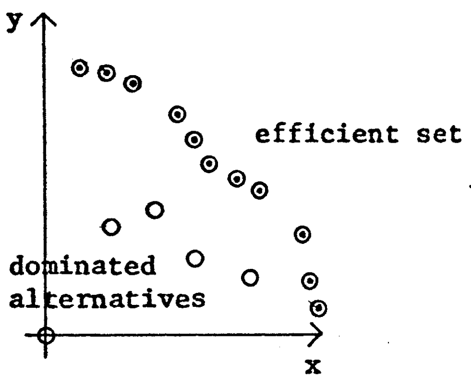


Figure No. 2.2

Dominated alternatives are eliminated from contention. If evaluators can assume continuous values (i.e. an infinity of alternatives is possible, as in linear programming), then the Pareto frontier will be convex (since an alternative lying on a straight line between any two alternatives on the frontier will be dominated by another alternative on the frontier. Convexity may not hold for a finite number of alternatives, however. In subsequent models, we assume that all alternatives being evaluated are on the Pareto-efficient frontier; thus higher preference models are required to resolve ambituities among this undominated set.

In the "Static Evaluation" model, the Pareto-efficient frontier is determined by constructing "quasi-levels", from directed graph theory. Alternatives are compared and ranked for each evaluator. For illustration, suppose that we have 7 alternatives, $A_1, A_2, \ldots, A_7$, being assessed with respect to 4 evaluators, w,x,y, and z. For example, with evaluator "w", we may have the ordering:

$$A_1 > A_2 > A_3 > A_4 > A_5 > A_6 > A_7 .$$

In directed graph form, this is represented as follows: (cf. Figure No. 2.3)

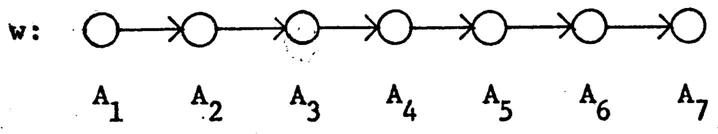


Figure No. 2.3

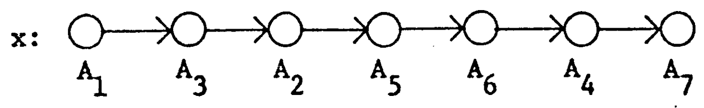
If for evaluators x, y, and z, we also have:

$$A_1 > A_2 > A_3 > A_4 > A_5 > A_6 > A_7,$$

then alternative $A_1$ completely dominates all others, and further, a complete ordering results. On the other hand, suppose that for evaluator y:

$$A_1 > A_3 > A_2 > A_5 > A_6 > A_4 > A_7,$$

which is represented as:



for evaluator y we have:

$$A_1 > A_3 > A_2 > A_6 > A_4 > A_5 > A_7$$

represented as:

y:
$$\begin{tikzcd} A_1 \arrow[r] & A_3 \arrow[r] & A_2 \arrow[r] & A_6 \arrow[r] & A_4 \arrow[r] & A_5 \arrow[r] & A_7 \end{tikzcd}$$

and for z, we have:

$$A_2 > A_1 > A_3 > A_4 > A_6 > A_5 > A_7,$$

represented:

z:
$$\begin{tikzcd} A_2 \arrow[r] & A_1 \arrow[r] & A_3 \arrow[r] & A_4 \arrow[r] & A_6 \arrow[r] & A_5 \arrow[r] & A_7 \end{tikzcd}$$

The overall ranking is derived from the combined directed graph, formed by including any line i,j if it occurs in any one of the directed graphs by evaluators: (cf. Figure No. 2.4)

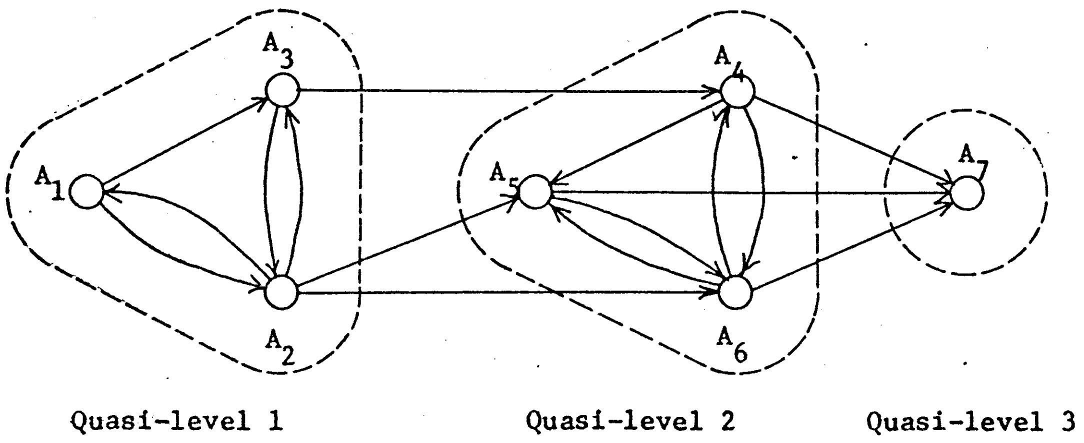


Figure No. 2.4

Alternatives $A_1$, $A_2$, and $A_3$ form an intransitive cycle, as do $A_4$, $A_5$, and $A_6$. The resulting quasi-levels are then:

| [1,2,3] | > | [4,5,6] | > | [7] |
|---|---|---|---|---|
| Quasi-level 1 | | Quasi-level 2 | | Quasi-level 3 |

Quasi-level 1 (consisting of 3 alternatives in this example) is the undominated set, or Pareto-efficient frontier. For large numbers of evaluators, the combined directed graph may be difficult to perceive or construct; then, the quasi-ordering can be obtained from the "reachability" matrix. (11) However, the probability of obtaining even quasi-orderings (apart from the Pareto frontier), goes down as the number of evaluators increases. Evaluation then operates on working out the intransitivities within quasi-levels, through improved measurement, or use of secondary evaluators.

## 2.3 Bounds on Preferences

Constraint levels put bounds on consequence values by dividing them into acceptable and unacceptable regions. Constraints must be used with caution, since they can be manipulated until only one, any one, or no alternatives remain in the acceptable consequence space. The resolution quality of constraints is low, and therefore boundaries between acceptable and unacceptable regions should not be

(11) F. Harary, R. Z. Norman, D. Cartwright; Structural Models: An Introduction to the Theory of Directed Graphs, (New York, N. Y.; John Wiley & Sons, Inc., 1965), p. 117.

treated as definitive, particularly at the beginning of the planning process. Figure No. 2.5 illustrates the use of constraints in particioning consequences into acceptable and unacceptable regions:

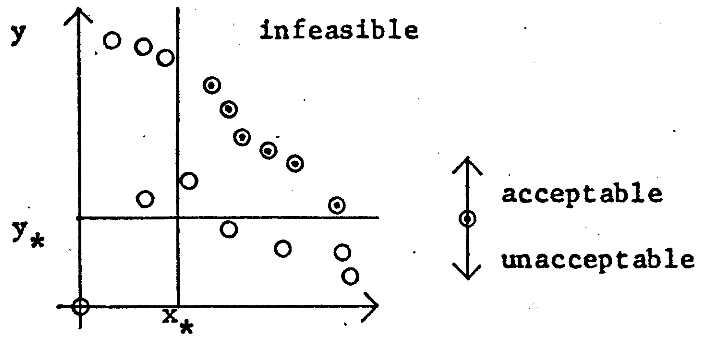


Figure No. 2.5

Constraints may define acceptable regions through both upper and lower bounds: (cf. Figure No. 2.6)

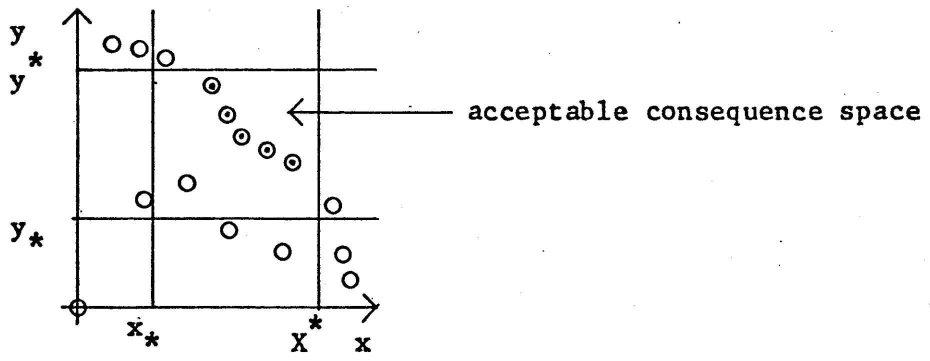


Figure No. 2.6

Even with preferences specified only to the degree of ordered consequences, and lower bound constraints, a number of choice procedures are possible, without requiring tradeoff information among evaluators:

(a) Satisficing Model: (12) set all evaluators with constraints, and choose the first alternative which satisfies them all (cf. Figure No. 2.7)

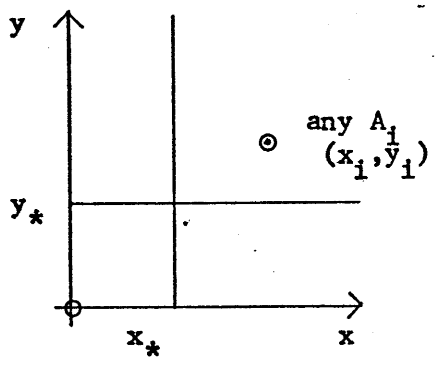


Figure No. 2.7

(b) Single Objective Maximization: set all but one evaluator with constraints, and select the alternative with the highest remaining evaluator: (cf. Figure No. 2.8)

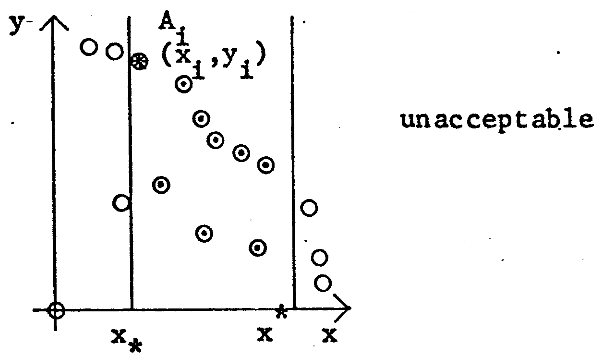


Figure No. 2.8

(12) H. A. Simon; "A Behavioural Model of Rational Choice", in Simon; Models of Man, (New York, N. Y., John Wiley & Sons, Inc., 1957).

The decision as to which evaluator will be left unconstrained may be arbitrary; although this can be alleviated somewhat by systematically loosening up each evaluator in turn, comparing the resulting selections, and choosing the alternative which appears most often.

The next set of models incorporate implied or explicit tradeoffs among evaluators, but risk and time preferences are still suppressed. All remaining alternatives under consideration lie on the efficient frontier. One of the simplest such preference structures is:

## 2.4 Lexicographic Ordering

Evaluators are ranked in order of their importance; e.g. evaluator x is more important than evaluator y. This principle pushes the preferred alternative towards lower (or higher) points on the efficient frontier: (cf. Figure No. 2.9)

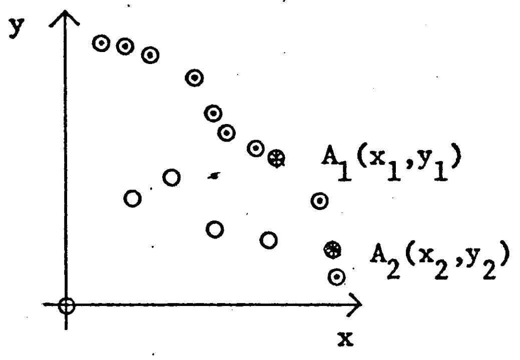


Figure No. 2.9

$A_2(x_2,y_2)$ will be preferred to $A_1(x_1,y_1)$ because $x_2 > x_1$. Only if $x_2 = x_1$, do we check for y values. Such a principle is generally not reasonable because no increase in one evaluator (y) can compensate for even a small decrease in a more preferred evaluator (x). However, over small ranges of evaluator values, it may be true. If the preference structure is lexicographic, indifference curves cannot be constructed since the decision-maker will never be indifferent between two distinct alternatives $A_1(x_1,y_1)$ and $A_2(x_2,y_2)$. If $x_1 \neq x_2$, then the alternative with the greater x value is preferred; if $x_1 = x_2$, the alternative with the greater y value is chosen.

The next set of models derives composite value functions by means of explicit tradeoff analysis. This is concerned with determining the rate of substitution of one evaluator with respect to another so that their combination may be represented by a composite function. The value function may be simplified if some form of value-wise independence among evaluators can be assumed from empirical testing.

Tradeoff analysis requires the use of indifference or iso-preference curves, which are defined by linking all (x,y) pairs to which the decision-maker is indifferent. The local substitution rate $\lambda$, at any point $(x_o,y_o)$ is the slope of the indifference curve through $(x_o,y_o)$: (cf. Figure No. 2.10)

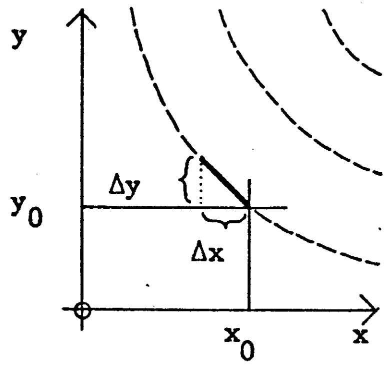


Figure No. 2.10

and $\lambda = \frac{\Delta y}{\Delta x}$ .

A value function V(x) reflects a decision-maker's preferences if:

$$x_1 \sim x_2 \longleftrightarrow V(x_1) = V(x_2)$$

and:

$$x_1 \succ x_2 \longleftrightarrow V(X_1) > V(x_2)$$

Four different preference models can be distinguished, in terms of their simplifying assumptions about the assessment of indifference curves, and the corresponding difficulty of analysis. The first three of these permit composite value functions to be computed; the last requires substantial empirical testing, and is only used if the problem context does not justify the use of the first three models:

(1) Constant linear indifference curves

(2) Constant form indifference curves

(3) Indifference curves with a constant rate of variation

(4) Complex indifference curves (not amenable to analytical forms).

## 2.5 Tradeoff Analysis: Constant Substitution Rate

If it can be determined that the substitution rate between evaluators x and y, at any point $(x_o, y_o)$ does not depend on the particular values $x_0$ and $y_0$, then the local substitution rate $\lambda$, is also the global substitution rate, and linear indifference curves of the form:

$$x + \lambda y = k \text{ (constant)}$$

result. The intersection of the Pareto-efficient frontier (which is convex in the continuous case) by the family of curves $x + \lambda y$, yields the most preferred alternative. (cf. Figure No. 2.11)

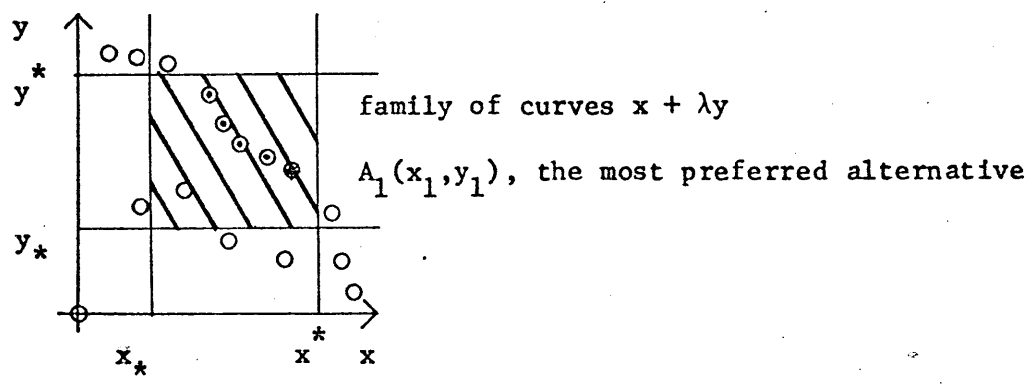

y
$y^*$
$y_*$
$x_*$
$x^*$
x
family of curves $x + \lambda y$
$A_1(x_1, y_1)$, the most preferred alternative

Figure No. 2.11

For a small number of alternatives, it may be sufficient to determine substitution intervals, such that: one alternative $A_1$ is preferred if $\lambda_1 > \lambda > \lambda_2$; another, $A_2$ is preferred if $\lambda_2 > \lambda > \lambda_3$; and so on. Since the substitution rate does not depend on the values of x and y, the two evaluators are considered to be "value-wise independent" (13) of each other. This assumption underlies models such as the Linear Scoring Function (*) and mathematical optimization techniques such as linear and separable programming. In this case, the composite value function takes the form:

$$V(x,y) = x + \lambda y.$$

## 2.6 Tradeoff Analysis: Constant Substitution Rate with One Transformed Variable

A slightly more complex form can be used if the local substitution rate at $(x_0,y_0)$ is found to depend on the value of one evaluator, say $y_0$, but not on the value of the other evaluator, $x_0$. (cf. Figure No. 2.12).

(13) M. L. Manheim & F. Hall; Abstract Representation of Goals, (Cambridge, Mass., M.I.T. Dept. of Civil Engineering, Professional Paper P67-24, January, 1968), p. 5.

(*) discussed in detail on page 102

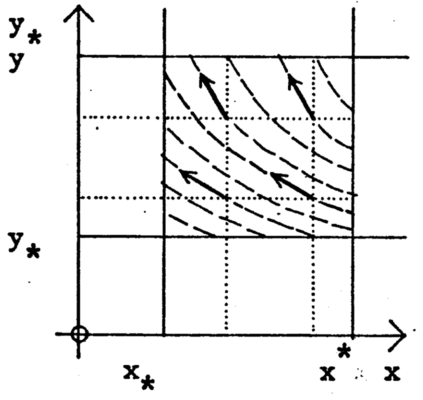


Figure No. 2.12

A composite value function which produces this pattern of local substitution rates, is:

$$V(x,y) = x + V_y(y),$$

where $V_y(y)$ is a global substitution function between x and y, assessed along any line $x_0$, $x_* < x_0 < x^*$: (cf. Figure No. 2.13)

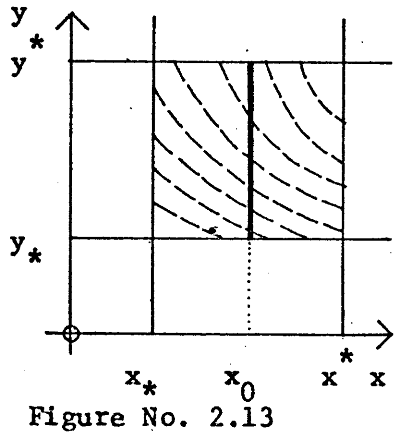


Figure No. 2.13

The $V_y(y)$ function may be thought of, as a rescaling or transformation of the evaluator y conditional on the interrelationship between x and y. Denoting a new variable $z = V_y(y)$, we illustrate in Figure No. 2.14, a possible relationship between z and y:

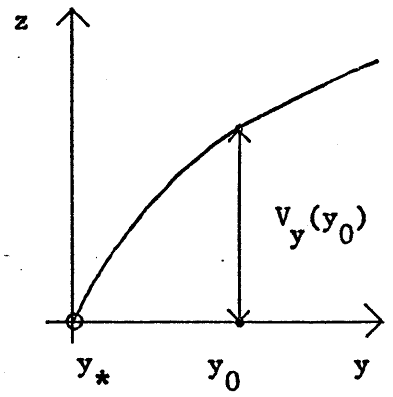


Figure No. 2.14

These assumptions may be made more useful if held to restricted values of x (or of y), and the analysis is repeated several times over the full range of either evaluator.

## 2.7 Tradeoff Analysis: Constant Substitution Rate with Two to N Transformed Variables

In general, the local substitution rate at any point $(x_0,y_0)$ will depend on the levels of both evaluators $x_0$ and $y_0$. However, it may still be possible to transform the x evaluator into a "w"-scale, and the evaluator y into a "z"-scale so that the local substitution rate at $(w_0,z_0)$ will not depend on the levels of $w_0$

or $z_0$; i.e. the transformed evaluators w and z are value-wise independent. To test whether this condition holds, we can attempt the "Corresponding Tradeoffs" test (14) which determines for any y held constant, at $y_0$ if the local substitution rate depends only on x values; and likewise, for any x held constant at $x_0$, if the substitution rate depends only on values of y. Figure No. 2.15 illustrates this test:

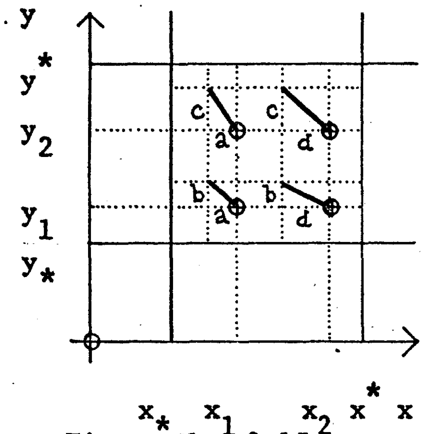


Figure No. 2.15

If the local substitution rates at $(x_1 y_1)$, $(x_2,y_2)$, $(x_1,y_2)$, and $(x_2,y_1)$ correspond as illustrated above, then the composite value function has the form:

$$V(x,y) = V_x(x) + V_y(y)$$

(14) H. Raiffa; "Tradeoffs under Certainty", (Cambridge, Mass., Harvard University, unpublished notes, 1968)

The $V_x(x)$ and $V_y(y)$ functions are plotted from a conjoint re-scaling procedure which:

(a) selects an arbitrary $x_m > x_*$, setting $V_x(x_m) = 1$;

(b) chooses $y_m$ so that $(x_m,y_*) \sim (x_*,y_m)$;
then $V_y(y_m) = 1$;

(c) continues to determine intermediate values from indifference relations; choosing $x_n$ and $y_n$ so that:
$(x_n,y_*) \sim (x_m,y_m) \sim (x_*,y_n)$;
then $V_x(x_n) = V_y(y_n) = 2$;
and so on, for $x_p$, $y_p$; $x_q$, $y_q$; etc.;

(d) fairs in resulting $V_x(x)$ and $V_y(y)$ curves. (cf. Figure No. 2.16)

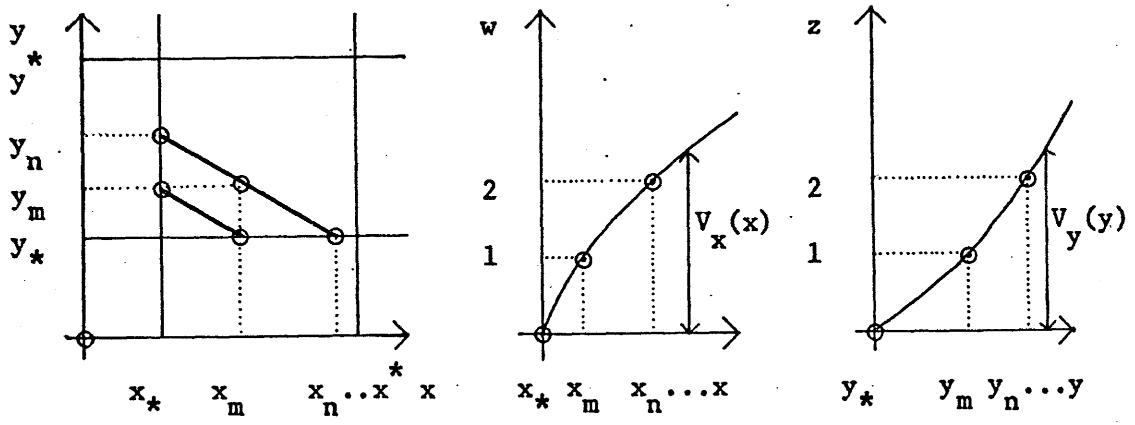


Figure No. 2.16

As with 2.6, the $V_x(x)$ and $V_y(y)$ functions may be thought of as monotonic rescalings or transformations of the evaluators x and y so as to reflect their mutual interrelationship. These transformed functions cannot be derived independently of one another. The same test may also be extended to n evaluators.

The Corresponding Tradeoffs Test requires constantly varying indifference curves for x and y. If it cannot be verified, then the $V_x(x)$ and $V_y(y)$ functions cannot be determined, and we are forced to a more detailed empirical analysis. Operationally however, it may be possible to accept the independence assumption over restricted ranges of x and/or y.

## 2.8 Complex Indifference Curve Analysis (15)

Indifference curve analysis must be used when the analytical form of the preference curves cannot be fitted, or where there are substantial interdependencies among evaluators, which if neglected, would lead to significant distortions. Indifference curves are derived by systematically comparing combinations of (x,y) evaluator pairs and determining preferences between the pairs. (cf. Figure No. 2.17)

(15) adapted from: K. R. MacCrimmon; Improving the System Design and Evaluation Process by the Use of Trade-off Information: An Application to Northeast Corridor Transportation Planning, (New York, N. Y.; Rand Corporation memo RM-5877-DOT, 1969).

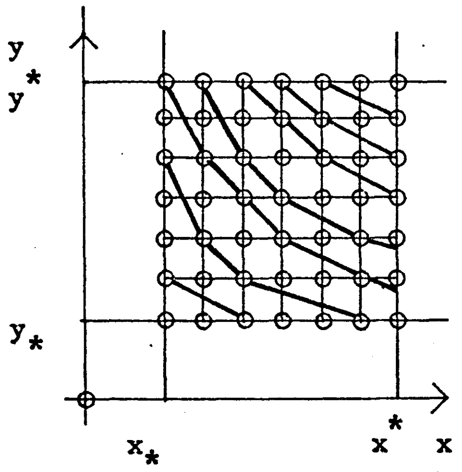


Figure No. 2.17

Indifference or "iso-preference" curves join all (x,y) pairs which are indifferent to one another in the decision-maker's value system. A significant disadvantage of the method is that the decision-maker is forced to explore a wide range of (x,y) combinations, only a few of which are likely to turn up in the various alternatives being considered. Further, for more than a few evaluators, the analysis is costly and time-consuming, since all pairs of evaluators must be examined. Therefore, the various value-wise independent models may serve as useful approximations to more complex preference interdependencies. This point is discussed in more detail in Section 3.

The final set of models continues tradeoff analysis, but introduces preferences for risk as developed in one-dimensional utility theory. Utility theory assumes that a continuous function exists for each preference dimension; this may be hard to justify empirically, since people would rather make real choices than define their preference curves through hypothetical lotteries.

The various multi-dimensional utility models require that one-dimensional utility functions for each evaluator be already assessed, or at least computable. A variety of techniques have been suggested to assess one-dimensional utility functions; a reasonable set of axioms such as that of von Neumann and Morgenstern (16) is presupposed. As with one-dimensional utility theory, rational decision-making under risk consists of picking the alternative with the highest expected (composite) utility.

## 2.9 Additive Utility

The additive value function representations of the preceding models, i.e.:

$$V(x,y) = V_x(x) + V_y(y)$$

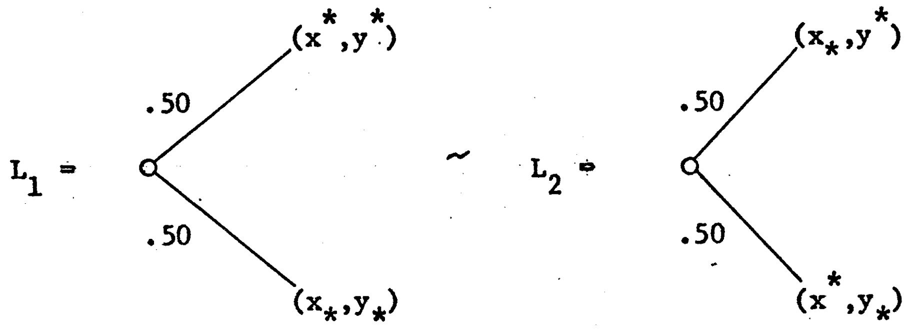
cannot be adapted directly to decision-making under risk, since value functions are appropriate only for certain consequences. The corresponding utility model, i.e.:

$$u(x,y) = u_x(x) + u_y(y),$$

requires in addition, the assumption that the desirability of any

(16) J. von Neumann & O. Morgenstern; op. cit.

lottery depends only on the marginal probability distributions of the consequence values, but not on their joint probability distributions. (17) This can be tested by determining if the decision-maker is indifferent between the following two lotteries: (assuming that $x^*$, $y^*$, $x_*$, and $y_*$ have already been assessed)



Each lottery has the same marginal probability distributions for x and y ordered consequences. Figure No. 2.18 represents these lotteries:

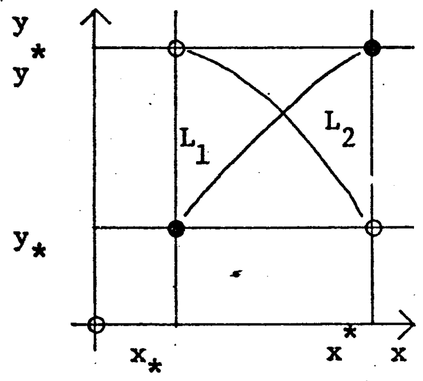


Figure No. 2.18

(17) P. C. Fishburn; "Independence in Utility Theory with Whole Product Sets", Operations Research, (Vol. 13, 1965), pp. 28 - 45.

By scaling $u(x_*,y_*) = u_x(x_*) = u_y(y_*) = 0$,

and $u(x^*,y^*) = u_x(x^*) = u_y(y^*) = 1$,

and defining: $u_x(x) = u(x,y_*)$,

$u_y(y) = u(x_*,y)$,

Fishburn derives the additive representation:

$$u(x,y) = u_x(x) + u_y(y).$$

However, Keeney notes:

"The main advantage to the additive utility function is its relative simplicity. The assessment of the n-dimensional utility function is reduced to the assessment of n one-dimensional utility functions, and as previously mentioned, adequate systematic procedures do exist for assessing one-dimensional utility functions. A major shortcoming of this approach is the restrictiveness of the necessary assumptions. We would often expect the utility of a lottery to be dependent not only on the marginal distributions of the respective attributes (evaluators), but also on their joint probability distribution." (18)

He goes on to develop the quasi-additive utility forms, which do not suffer from this restriction.

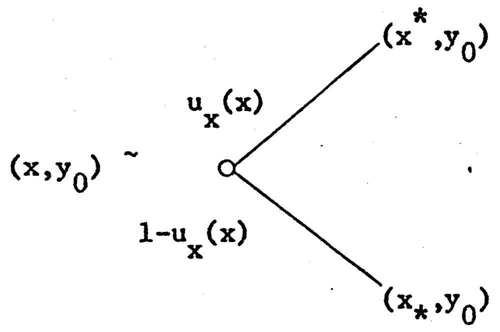
## 2.10 Quasi-Additive Utility

The simpler form of this representation requires evaluator y to be utility independent of x, and x to be utility independent of y.

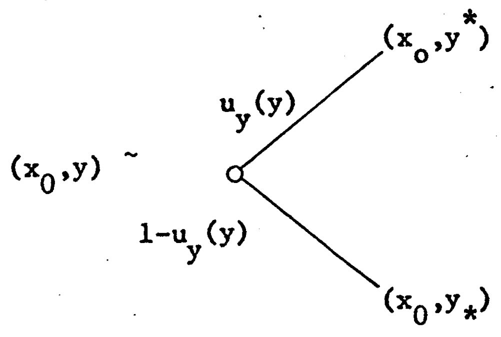
(18) R. L. Keeney; Multidimensional Utility Functions: Theory Assessment and Application, (Cambridge, Mass., M.I.T. Operations Research Center, Technical Report No. 43, Oct., 1969), p. 25.

This implies that for a given value of one of the evaluators, say $y = y_0$, the compound utility function $u(x,y_0)$ will depend only on a function of the x values; similarly, for $x = x_0$, the function $u(x_o,y)$ will depend only on a function of the y values. The joint utility function $u(x,y)$ is derived in four steps:

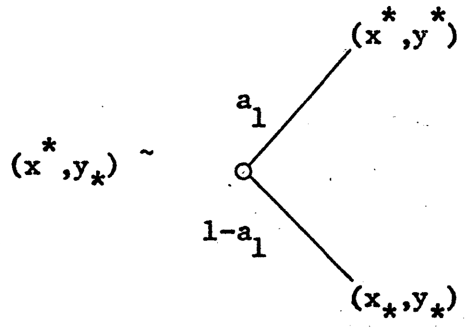
(1) Since x is utility independent of y; for any $y_0$, $(y^* > y_0 > y_*)$, we can define a conditional utility function on x, $u_x(x)$, so that:



(2) Since y is utility independent of x; for any $x_0$, $(x^* > x_0 > x_*)$, we can define a conditional utility function on y, $u_y(y)$, so that:

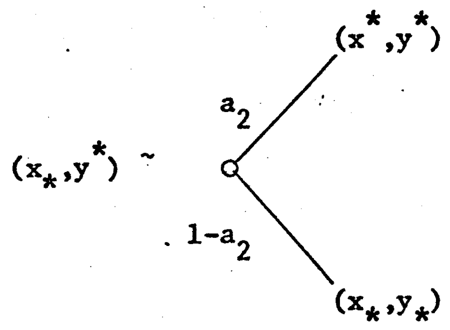


(3) Determine the value $a_1$, so that:



i.e., $a_1 = u(x^*,y_*)$

(4) Determine the value $a_2$ so that:



i.e., $a_2 = u(x_*,y^*)$

The compound utility function for two evaluators, is then:

$$u(x,y) = a_1u_x + a_2u_y(y) + (1 - a_1 - a_2)\ u_x(x)u_y(y)$$

Figure No. 2.19 shows a graphical interpretation of this result:

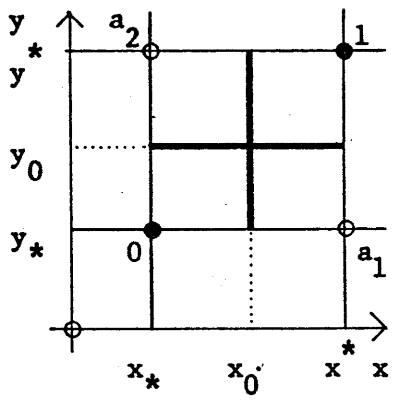


Figure No. 2.19

The compound utility for any point in the acceptable consequence space is uniquely determined by the relative utilities of consequences along $y_0:(u_x(x))$, and along $x_0:(u_y(u))$, and the two points, $a_1:(u(x^*,y_*))$, and $a_2:(u(x_*,y^*))$. What must be assessed are the two utility functions represented by the heavy lines in the diagram, and the two circled corner utility points.

Note that if $a_1 + a_2 = 1$, then the additive utility form results. Therefore, the quasi-additive procedure should be adopted generally, and if $a_1 + a_2 = 1$, then the simpler additive form will result anyway.

## 2.11 Asymmetric Quasi-Additive Utility

The more complex form of tuility independence requires at least one evaluator, say x, to be utility independent of the other, y; but not vice versa.

The procedure uses:

(1) Step (1) above, to derive $u_x(x)$;

(2) Step (3) above, to derive $a_1$;

(3) Step (4) above, to derive $a_2$;

(4) asseses a function $u(x_*,y)$, by getting p values, so that:

$$\begin{tikzcd}
 & & (x_*,y^*) \\
(x_*,y) \sim & \circ \arrow[ur, no head, "p"] \arrow[dr, no head, "1-p"'] & \\
 & & (x_*,y_*)
\end{tikzcd}$$

i.e., $u(x_*,y) = pa_2$;

(5) assesses a function $u(x^*,y)$ by getting s values, so that:

$$\begin{tikzcd}
 & & (x^*,y^*) \\
(x^*,y) \sim & \circ \arrow[ur, no head, "s"] \arrow[dr, no head, "1-s"'] & \\
 & & (x^*,y_*)
\end{tikzcd}$$

The following compound utility results:

$$u(x,y) = u_x(x)u(x^*,y) + (1 - u_x(x))u(x_*,y)$$

Figure No. 2.20 shows graphically, the assessments which must be made:

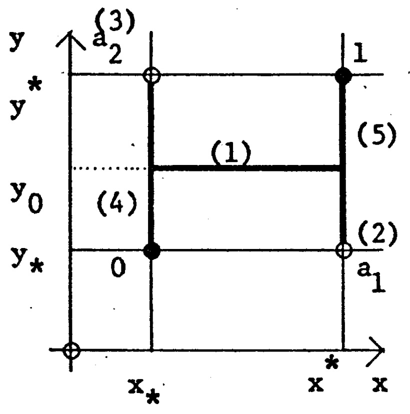


Figure No. 2.20

Keeney (19) also discusses cases in which one of the conditional utility functions may be replaced by an iso-preference or indifference curve; or two of the conditional utility functions are replaced by two indifference curves. The assumption of utility independence is also useful as an approximation even if not all evaluators are utility independent of one another. In such cases, the representation can be simplified by grouping the evaluators into two or more utility-independent vectors; and using the degrees of freedom inherent in the quasi-additive form to fit empirically, the conditional utility functions.

Preferences for time (i.e. when a given consequence occurs) are not considered here. In this section, we have examined a number of preference models arrayed in order of precision of

(19) Ibid.; pp. 59 - 65.

measurement. At one extreme, we have the simple ordering of consequences; at the other, complex multi-dimensional utility theory. Any of the preference models may be used as part of a larger evaluation strategy, though ordinarily, it would be expected that simpler techniques would be applicable in the early stages of the planning process, and more complex techniques would be applied to only a few alternatives (about which there is genuine ambiguity), later on in the process. For example, the proposed "Static Evaluation" model (*) first uses a simple check for dominance among alternatives, deletes dominated alternatives, and then applies one of a set of more detailed preference models, for selection among the remaining alternatives. Available techniques then include the additive value, additive utility, and quasi-additive utility models outlined above, as well as two "relative value" preference models applicable only to a fixed set of alternatives.

Multi-dimensional preference models are also relevant to the discussion in Section 3, of hierarchical goal models. There, we describe in more detail, the relationships between general, aggregate goal variables, and multi-dimensional disaggregated evaluators; relationships which are usually arrayed in the form of a goal hierarchy. The issues of independence vs. interdependence among evaluators, crucial to hierarchicization, are

(*) see page 102

also considered. The resulting conclusions underly the approach taken in both the "static" and "Dynamic Evaluation" models described in Section 4.

## 3. HIERARCHICAL SYSTEMS MODELS

At the core of planning, particularly for urban activity systems, is the simplification or abstraction of complex empirical reality for the purposes of control. Behind this striving for simplicity, lie two central factors:

(1) our limited information handling and computational abilities, which inhibit our understanding of complex systems;

(2) the redundancy present in most complex structures; a factor which can be used to simplify our descriptions of them.

In terms of (1), the "computation" issue, complicated problems can usually be solved only by dividing or decomposing them into a number of parts, each of which can be attacked by a smaller search effort. Minsky states:

> "Generally speaking, a successful division (of a complex problem) will reduce the search time not by a mere fraction, but by a fractional exponent. ...thus, practically any ability at all to "plan" or "analyse" a problem will be profitable, if the problem is difficult." (20)

In terms of (2), the "representation" issue, Simon argues that the perceived complexity or simplicity of a system depends as much on our description or representation language, as on the objective complexity of the system; the problem being to find a representation which will eliminate most of the redundancies of

(20) M. A. Minsky; "Steps Toward Artificial Intelligence" in E. A. Feigenbaum & J. Feldman, eds.; Computers and Thought, (New York, N. Y., McGraw-Hill, 1963), p. 442.

the empirical structure:

> "....one path to the construction of a non-trivial theory of complex systems is by way of a theory of hierarchy. Empirically, a large proportion of the complex systems we observe in nature exhibit hierarchic structure. On theoretical grounds we could expect complex systems to be hierarchies in a world in which complexity had to evolve from simplicity. In their dynamics, hierarchies have a property, near decomposability, that greatly simplifies their behaviour. Near decomposability also simplifies the description of a complex system and makes it easier to understand how the information needed for the development or reproduction of the system can be stored in reasonable compass." (21)

From the perspective of the planning process, there is a good deal of intuitive justification therefore, for the hierarchical factoring of particular problem spaces. Factoring of general goals into multi-dimensional objectives serves as an approximation for goals which cannot be measured in practice. Factoring of solution spaces into different levels of description, permits simplification in that certain kinds of information can be included with each solution level, that may be reasonably omitted or approximated at other levels. However, consistency of decomposition is difficult to maintain from problem context to problem context. Tests of "reasonableness" of application rather than formal or theoretical rules, apply in this area.

However, there have been some tentative steps towards formalizing the theoretical bases of hierarchical systems: one

(21) H. A. Simon; The Sciences of the Artificial, (Cambridge, Mass., M.I.T. Press, 1969), p.

fundamental work is that of Mesarovic et al (22). They define a multi-level hierarchical structure as:

> "...a vertical arrangement of subsystems which comprise the overall system, the priority or right of intervention of the higher level subsystems, and the dependence of higher level subsystems upon the actual performance of the lower levels."

Figure No. 3.1 illustrates this concept graphically. The term "system" refers to a transformation of input data into outputs.

Vertical Interaction between Levels of a Hierarchy

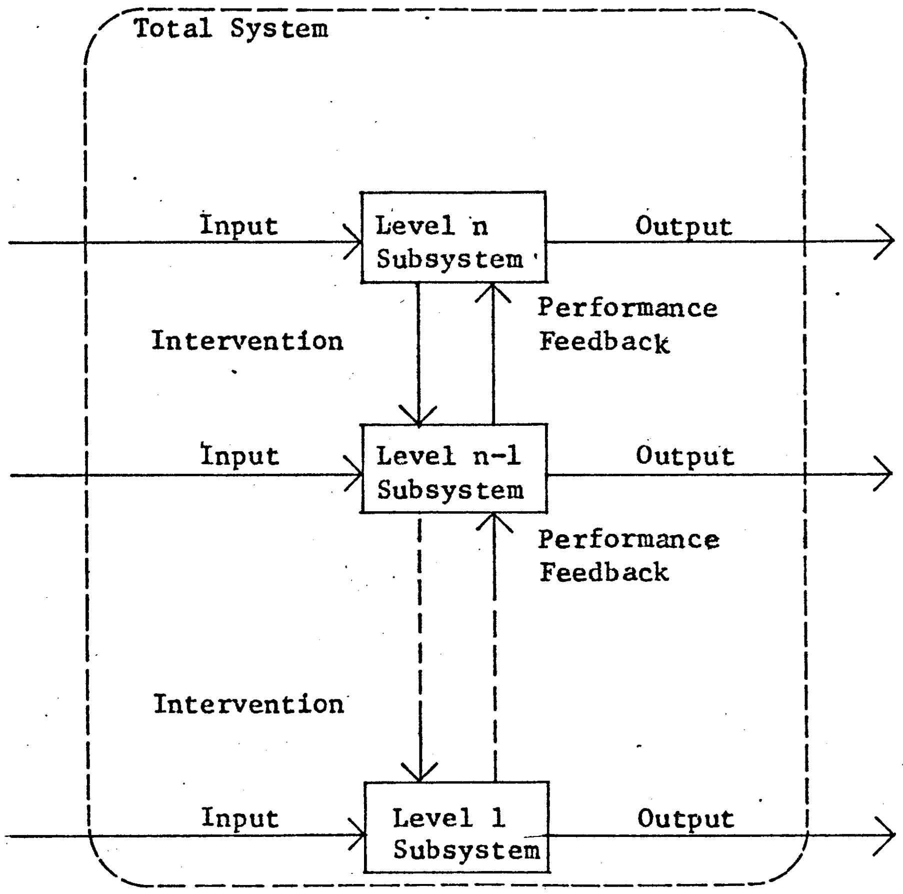


Figure No. 3.1 (23)

(22) M. D. Mesarovic, D. Macko, Y. Takahara; Theory of Hierarchical, Multilevel Systems, (New York, N. Y., Academic Press, 1970), p. 34.

(23) Ibid.; Figure 21, p. 35.

The crucial task in defining a hierarchical system, is the designation of system levels. Mesarovic notes that:

> "(i) there is an order of magnitude difference in the size of the units of concern on different levels.
>
> (ii) what constitutes a unit on a particular level depends on the interaction mechanisms operative in that particular level...." (24)

Simon's concept of "nearly-decomposable systems (25) is similar: the interactions among subsystems are weak, but not negligible. At any particular level in the system, the weak interactions among subsystems are distinguishable from the stronger interactions within the subsystems. The former are of different orders of magnitude at different system levels. These criteria do not have meaning however, outside of the context of a particular decision problem. More useful for our purposes, is Mesarovic's distinction of three functional types of levels: (26)

(1) "strata" (levels of description or abstraction);

(2) "layers" (levels of decision complexity);

(3) "echelons" (organizational levels).

In defining strata or description levels, a balance must be struck between simplicity or economy of description, and the need to include as many system variables as are relevant to decisions

(24) Ibid.; p. 31.

(25) Simon; op. cit. (1969).

(26) Ibid.; p. 37.

at that level. Understanding of a system increases by crossing strata: in moving down the hierarchy, one obtains a more detailed explanation; while moving up the hierarchy, one obtains a deeper understanding of its significance.

In defining layers or decision levels, the balance is between the need to make a decision by a specified deadline, and the desire to understand the problem more clearly. A hierarchical structuring of decision layers defines a set of sequential decision problems, and a control procedure for solving them: the solution of a problem in the sequence, determines some of the inputs necessary for successive problem solutions; the overall problem is solved once all of the subproblems are solved.

The definition of echelons or managerial levels does not concern us here. In this sense, hierarchical levels serve as a formal vehicle for communication and transfer of control in an organization. Echelons must balance between information-handling overload implied by centralized, unitary control; and the lack of co-ordination implied by decentralized units.

There is by no means a necessary one-to-one correspondence between any of the functional types of levels: various combinations of strata and layers may occur in multiple-echelon systems; various interrelations of strata and layers for a unitary decision-maker, are possible. However, all three concepts of hierarchy have several principles in common (27):

(27) Ibid.; p. 54.

(1) a higher level unit is concerned with broader aspects of overall system behaviour;

(2) the decision period, or time horizon, of higher levels, is longer than for lower levels;

(3) higher levels are concerned with the slower aspects of overall systems behaviour;

(4) descriptions and problems on higher levels are less structured, with more uncertainties, and are more difficult to formalize quantitatively, than lower levels.

In the evaluation model to follow, we make a distinction between hierarchically-structured goals in the evaluation language, and hierarchically-structured actions in the action space. The latter corresponds to Mesarovic's "strata" or levels of description; the former to "layers" of decision complexity. Miller, Galanter and Pribram make the same distinction in their theory of behaviour (28), where the fundamental unit is the cybernetic feedback loop, or "TOTE" (test-operate-test-exit) pattern: (cf. Figure No. 32).

(28) G. A. Miller, E. Galanter, and E. H. Pribram; Plans and the Structure of Behaviour, (New York, N. Y., Holt, Rinehart & Wilson, Inc., 1960), p. 26.

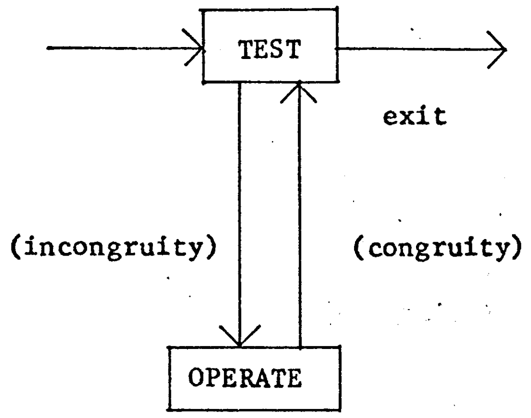


Figure No. 3.2

TOTE units can be chained in sequence, or may form operational components of larger TOTE units in the hierarchical organization of behaviour. Although they make a distinction between explorations in action space (operations) and mechanisms by which actions are tested for suitability (tests), there is always a one-to-one correspondence between tests and actions in the TOTE pattern. At the finest level of scrutiny, we would expect this correspondence to hold; however, in terms of our model which formalizes only "global" evaluation procedures, and leaves "local" tests embedded within search procedures, this would not always be true. For reasons of computational simplicity though, it is more convenient to ensure that evaluation levels and action levels coincide.

The units at each goal level (goals) and the units at each action level (designs) will, in general, not correspond or map directly onto each other. As quoted above:

"...what constitutes a unit on any particular level, depends on the interaction mechanisms operative in that particular level." (29)

Since we have constructed languages for two different purposes, the interaction mechanisms between components in the separate domains will also be different, and therefore, their respective units may not coincide. (*) The usual intent with factoring global goals into subgoals, is to get a set of multiple objectives as disjoint or as independent from one another as possible. The hierarchical stratification of actions, however, results in a set of overlapping regions or action spaces, within the region of including higher-level actions. Models which attempt to integrate the two languages, such as Alexander's hierarchical decomposition (30), are not entirely successful in this regard. Manheim states:

"The underlying issue here is complex. There is one language in which we naturally describe actions, and there is another which expresses our evaluations of those actions. Our natural tendency is to aggregate actions within the frame of reference provided by the descriptor language. But in order to get high similarity among actions, we want to aggregate them with regard to the evaluation language. The work of Christopher Alexander (Notes on the Synthesis of Form, Cambridge, Mass., Harvard University Press (1964)) can be described as a way of developing new descriptor languages such that there is

---

(29) Measrovic et. al.; op. cit., p. 31.

(*) some operational reasons for this are described, beginning on page

(30) Christopher Alexander; Notes on the Synthesis of Form, (Cambridge, Mass., Harvard University Press, 1964).

a greater correspondence between the descriptor and evaluation languages. In our terms, such a reworking of the descriptor language would yield a new metric, or set of metrics, on the action space." (31)

In the next section, we outline a number of models for hierarchical goal structuring and hierarchical planning; discuss and criticize each model, and suggest from that, some of the reasoning behind the particular approach we have taken.

## 3.1 Hierarchical Goal Models

Goals generated by, or assigned to, a planner, are usually multidimensional. Emery (32) suggests three reasons for this:

(1) compression of several incommensurable goals into a single objective, reduces their information content, unless there is an agreed-upon tradeoff between goals, which is acceptable for lower-level planning and control;

(2) multiple, measurable objectives can serve as approximations for more general goals which are not measurable operationally;

(31) M. L. Manheim; Hierarchical Structure (Cambridge, Mass.; M.I.T. Press, 1966), p. 157.

(32) J. C. Emery; Organizational Planning and Control Systems: Theory and Technology, (New York, N. Y.; MacMillan Co., 1969), p. 115.

(3) multiple goals form a useful means of conveying sufficient information about desired behaviour in the face of interaction effects among unpredictable context and design variables.

Global goals are made operational only after they have been factored into a hierarchy of sub-goals. Each sub-goal generated by this process may give rise to lower-level planning, which in turn, may generate still lower level goals as a means of achieving its own goals. The lowest level of goals is a set of performance criteria whose values can be assessed for each alternative under consideration. All of the hierarchical goal models to be discussed, follow similar reasoning.

(a) Goal Fabric Analysis (33)

The Goal Fabric model has two stages:

(1) An analytical phase; in which all the known goals are listed for the project, and then, the various relations among the goals are identified.

(2) A ranking of alternatives; in which each new alternative is mapped onto the goal fabric, compared with one previously ranked alternative, and fitted into the overall ranking. Only two alternatives at a time are evaluated.

(33) Manheim & Hall; op. cit.

Phase (1), goal structuring, determines only those relationships which are relevant to evaluation. Relations guide the expansion of the goals list in order to clarify the vague, general statements by which the problem may have been originally defined. Relations between goals may be of four kinds:

(1) Specification: the lower-level goal explains in more detail, a general goal;

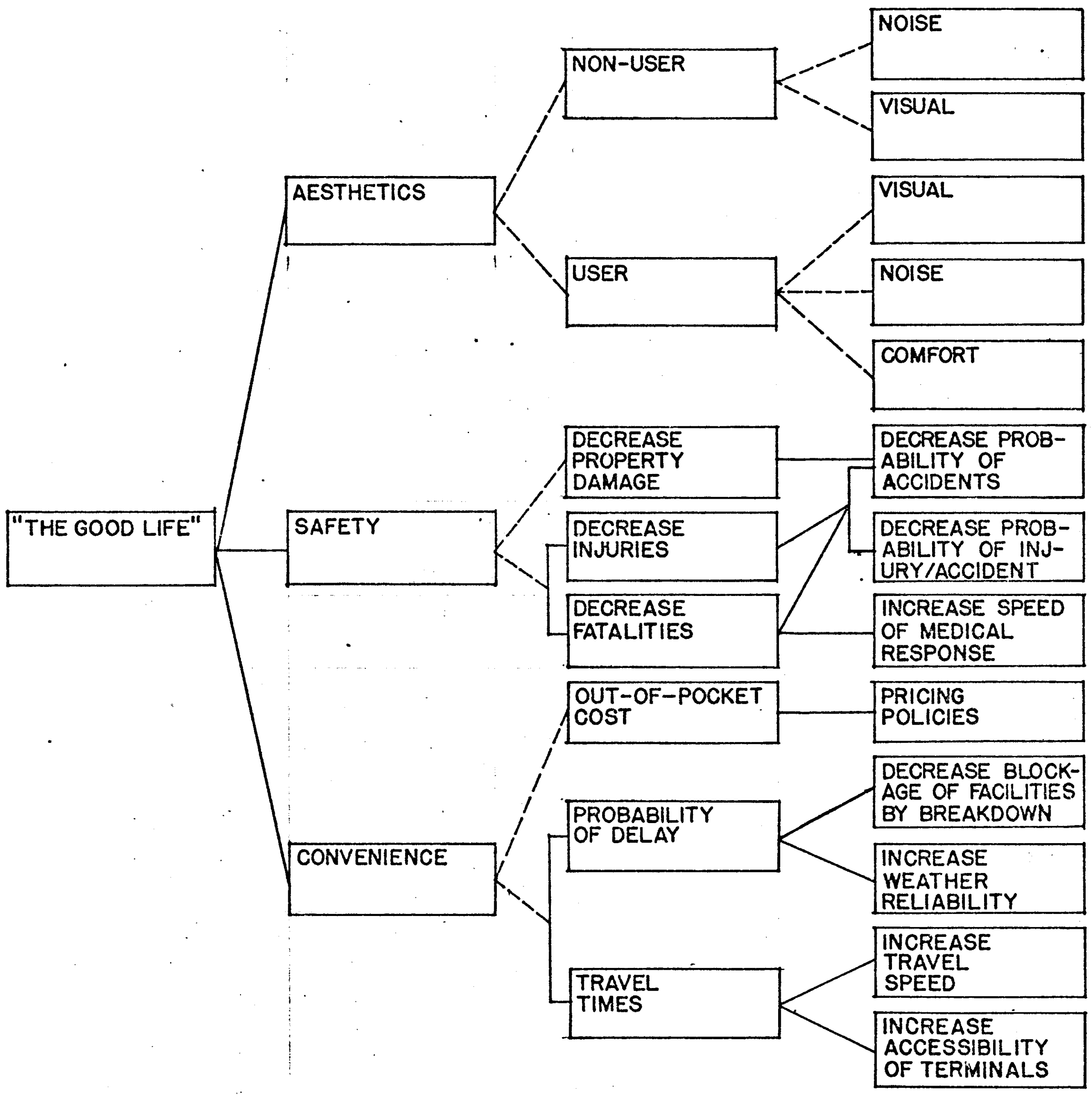
(2) Means-ends: the lower level goal explains how a general goal will be accomplished. The means goal is important only because it is instrumental in achieving an end;

(3) Value-wise dependence: denotes a goal which can only be evaluated in conjunction with other goals;

(4) Value-wise independence: denotes a goal which can be evaluated independently.

Once these relations are established and listed, they yield a hierarchical tree (cf. Figure No. 3.3) in which the lowest goals should be measurable. Evaluation first entails the mapping of alternatives onto the goal fabric: i.e. predicting the performance of alternatives with respect to goal subsets. Dominance checks are then applied to these subsets: if there is dominance over the set, then it can be transferred to the more general, upper-level goal; if there is not, combinations and tradeoffs among goals must be used, to determine which alternative dominates. Manheim and Hall suggest five techniques which are available at this point:

GOAL FABRIC ANALYSIS



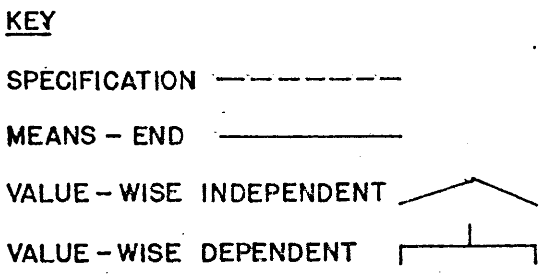


FIGURE NO. 3·3

(1) Dominance;

(2) Explicit choice by the decision-maker (essentially arbitrary);

(3) Comparison of intervals: find the interval between alternatives on each goal, then decide how these intervals compare with each other;

(4) Indifference measures: for example, alternative A is preferred over a certain goal variable range, and alternative B over another range; the actual goal values determine which alternative is selected;

(5) Modified Utility Measure: a simplified linear scoring function.

It is certainly possible to agree with the authors' objective of not forcing detailed assessment of the decision-maker's preferences unless absolutely necessary; in fact, this is the principal advantage of their model. However, once one of the techniques for assessing more detailed preferences is invoked, in order to clear up the ambiguity between two alternatives, then this assessment is available for any subsequent comparison. Should a large number of such ambiguities turn up in the evaluation procedure, the Goal Fabric model in practice requires the detailed preferences it was trying to avoid. As the number of problem dimensions increases, so does the probability that rough rankings or imprecise value statements will be conclusive for choice.

Once one alternative is preferred in some dimensions, and its competitor is preferred over the remainder, then a good deal depends on fortuitous structuring of the problem, or the ad-hoc procedures suggested, for a definitive selection to be made.

The reason for this quandry, is that decidability is related to the weakest form of preference measurement in a decision context. If the weakest form of measurement is an ordinal ranking, for some goal variable, then dominance for all goal variables can be assessed by constructing quasi-levels.* If there is still ambiguity (i.e. more than one alternative in each quasi-level), then the resolution among the alternatives within each quasi-level must be made by invoking a higher level of preference measurement (for non-arbitrariness). Given the von Neumann-Morgenstern theorems for constructing utilities from lottery comparisons (34), an interval utility scale can be constructed for preference judgments of any higher order than ordinal. The Goal Fabric model is ambiguous rather than systematic about when such judgments may have to be made.

---

*as discussed for example, on page no. 24

(34) von Neumann & Morgenstern; op. cit.

## (b) Miller's Additive Worth Hierarchy

James R. Miller (35) proposes a model for the assessment of "worth" values, in which goals are factored hierarchically into subgoals, these subgoals into further subgoals, and so on, until a level of detail is reached where physical measurement can be associated with each evaluator. His procedure is as follows:

(1) List the main performance objectives, which should be complete and exhaustive, mutually exclusive, worth-independent, and non-redundant.

(2) Generate a hierarchical structure of performance criteria.

(3) Select physical performance measures in the descriptor language, one for each lowest level performance criterion.

(4) Establish worth relationships between the lowest-level performance criteria and their associated physical performance measures. (i.e. "score" each alternative with respect to each eavluator).

(5) Establish a weighting or trade-off procedure for combining worth scores, to arrive at a single overall index of worth.

---

(35) J. R. Miller III: "The Assessment of Worth: A Systematic Procedure and Its Experimental Validation", (Cambridge, Mass., M.I.T. Sloan School of Management, unpublished Ph.D. thesis, 1966).

Figure No. 3.4 illustrates the hierarchical goal tree for an example from Miller's thesis: the selection of a job by a recent college graduate. The subject broke his goals into four major areas: monetary compensation, geographical location, travel requirements, and nature of work. Each of these is further subdivided: for example, "monetary compensation" is broken down into "immediate" and "future"; "immediate" into "starting salary" and "fringe benefits"; "fringe benefits" into "retirement" and "insurance".

The weights or tradeoff values assigned to each level sum to 1.0. At the lowest performance level, the weight assigned to an evaluator is the product of the level weights assigned to its direct chain of "parent" or including goals. Thus, "retirement" (fringe benefits) receives a weight of:

$$\lambda_3 = .33 \times .70 \times .10 \times .40 = .009$$

The overall worth of a set of performance evaluators $e_1, e_2, \ldots, e_n$, is found by multiplying each worth score by its associated performance tradeoff weight:

$$W(e_1, e_2, \ldots, e_n) = \sum_{i=1}^{n} \lambda_i W_i(e_i),$$

where $\lambda_i$ represents the tradeoff rate for evaluator $e_i$, and:

$$\sum_{i=1}^{n} \lambda_i = 1.0$$

MILLER'S ADDITIVE WORTH HIERARCHY

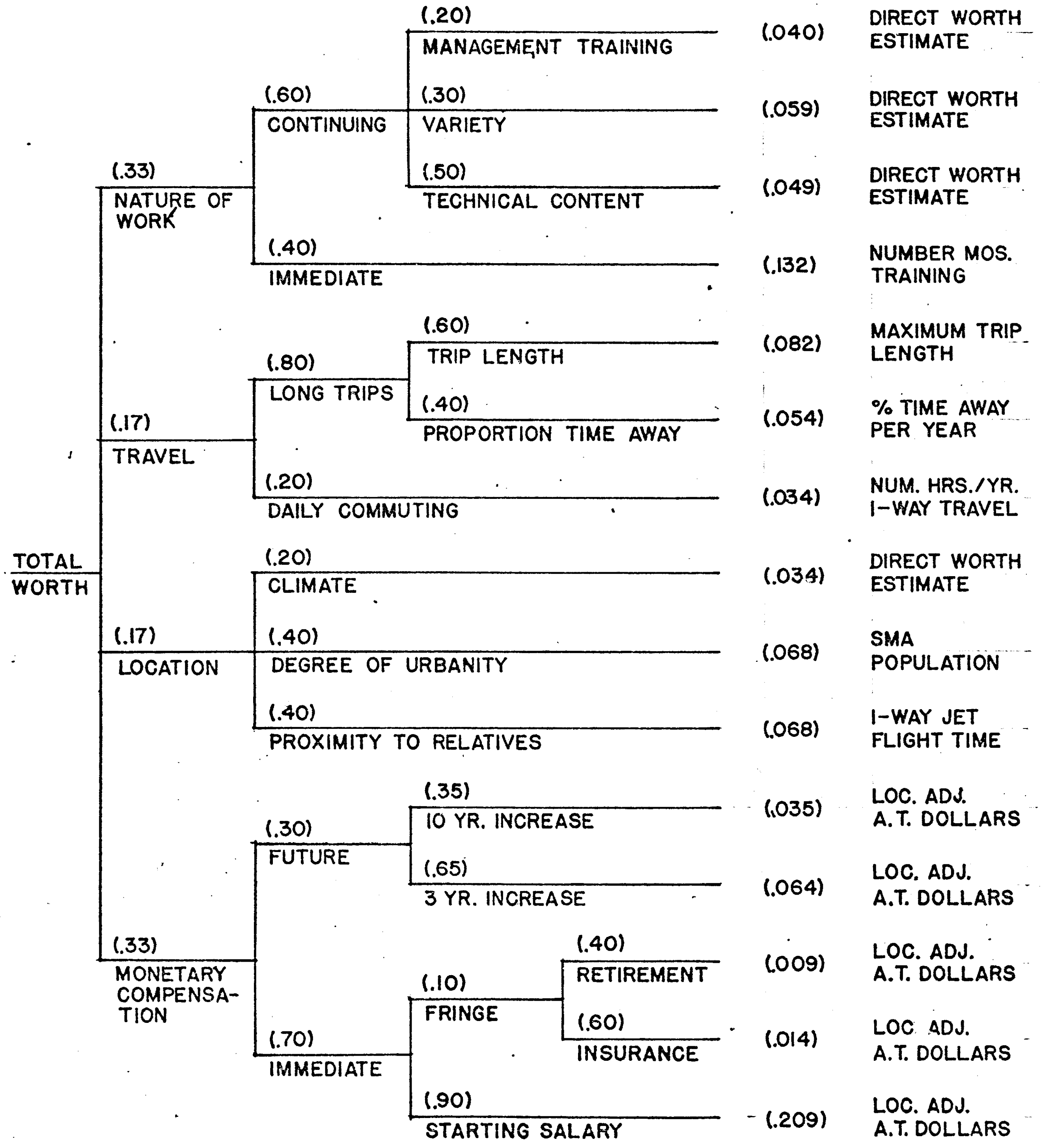


FIGURE NO. 3·4

The $W(e_i)$ values must all be consistently scaled in the interval between 0 and 1.

The crucial requirement for a simple aggregation of worth values, is the concept of "worth independence" which implies a substitution rate between evaluators, constant for all values that these evaluators may take on. (*) Miller outlines a procedure for eliminating worth independence if it occurs: goal variables are eliminated, redefined, or combined with other goal variables, to ensure independence. Worth independence is also essential to additive utility models; it ensures computability and thereby, decidability. By assuming independence among goal variables, Miller's procedure is able to derive an unambiguous overall total worth every time. On the other hand, Manheim and Hall (36) cannot guarantee unambiguous dominance in every comparison of alternatives since they permit "value-wise dependence", but operationally they require less preference information, and computation from the decision-maker.

Miller's worth concept is not applicable to probabilistic outcomes since it does not measure aversion to risk. The extension of the additive worth concept to risky decision problems, i.e.:

---

(*) see constant substitution rate with linear indifference curves, p. 33

(36) Manheim & Hall; op. cit.

$$u(e_1, e_2, \ldots e_m) = \sum_{i=1}^{m} u_i(e_i)$$

requires, in addition, the marginality assumption mentioned earlier.*

## (c) Means-Ends Analysis

Means-ends analysis may be viewed both as a procedure for decomposing a goal tree, and as a sequential decision process (as implemented in Newell, Shaw, & Simon's computer program, GPS (37)). In terms of goal decomposition, it is also included within both the Manheim & Hall, and Miller hierarchical goal models described above, as a component; although both of these models also permit other relations among goal variables. Means-ends analysis divides overall problem objectives into a set of subgoals instrumental to achieving these objectives. The subgoals in aggregate specify what is meant by their parent objective; they are important to the decision-maker only as intermediate steps to satisfying these ends. Normally, all subgoals must be satisfied before considering the parent objective fulfilled. The

(*) cf. page no. 42

(37) A. Newell, J. C. Shaw, & H. A. Simon; "A general problem-solving program for a computer," Computers and Automation, (Vol. 8, No. 7, 1959), pp. 10-16.

subgoals in turn can be considered as "ends", each of which can be satisfied by further decomposed "means" subgoals. The process continues to a level of detail where the performance of alternatives with respect to means subgoals can be assessed or measured; the overall satisfaction for each alternative is computed by aggregating means worth values through the chains of means-ends "staircases". Figure No. 3.5 illustrates a portion of a means-ends analysis for a business firm choosing between specialized or combined district managers: (38)

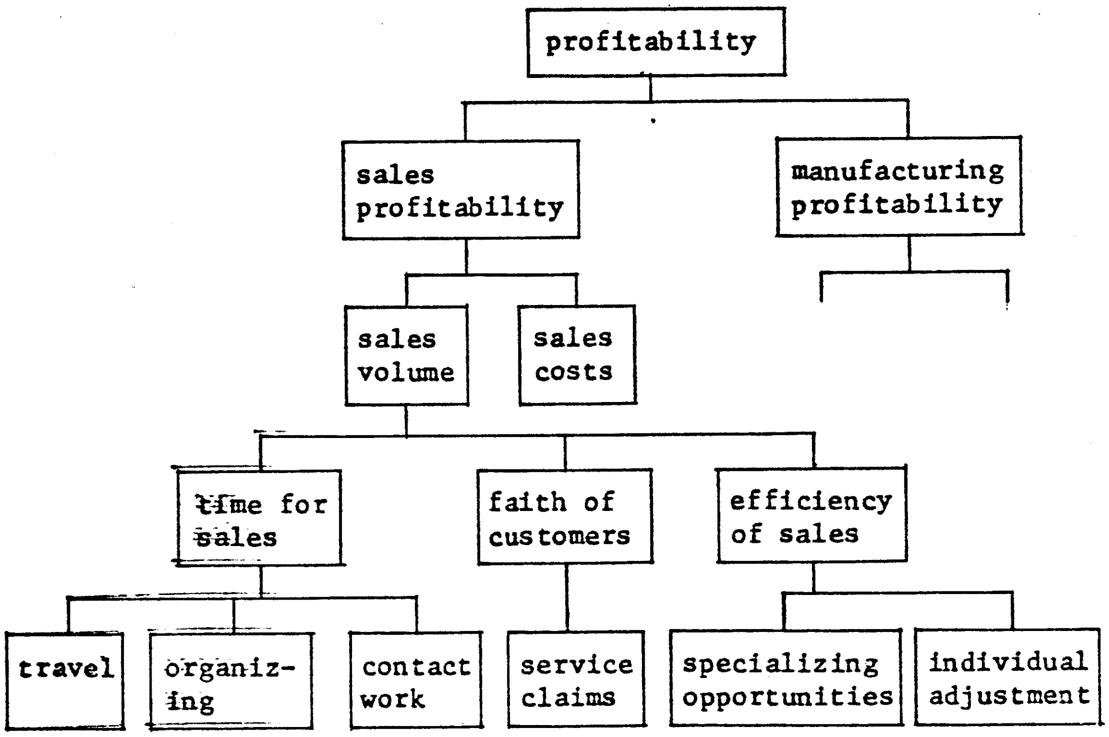


Figure No. 3.5

(38) E. Johnsen; Studies in Multi-Objective Decision Models, (Lund, Studentlitteratur, Economic Research Center in Lund, Monograph No. 1, 1968), p. 260.

The decomposition results in a goal tree with no overlapping staircases; i.e. one subgoal cannot serve as a "means" to more than one "ends" goal. The greatest difficulty in the method (or any goal fabric method) lies in determining whether each set of means subgoals is complete in the sense of defining satisfaction for their parent end goals. The relative contribution of each subgoal to its parent goal is also an issue in the aggregation of performance measures for an alternative. In this respect, means-ends analysis shares the same problems as Miller's goal hierarchy discussed above.

In its mechanistic form, as a sequential decision model, means-ends analysis is more interesting, since it bridges the gap between hierarchical goal structures and planning models, albeit in a simplistic manner. The logic of Newell, Shaw, & Simon's computer program GPS, for example, is recursive: given the present set of goals, it attempts to solve the problem from its given repertoire of operators; if the problem is insoluble, the present set of goals is decomposed into a set of subgoals, and the procedure calls itself again, as a subroutine. The problem is decomposed only to the point where its subproblems can be solved; the aggregated solution to all subproblems at all goal levels, defines a solution to the problem. GPS can only deal with well-defined problems which have all goals specified as constraints: even in this framework, a lot of backtracking and traversing of not aggregate into a solution to a higher level sub-problem.* Alexander's hierarchical decomposition model (39) does not even consider this possibility: the diagramming phase (left unformalized in that model) assumed that all sub-problem solutions would be compatible, and could be meshed with each other in the final solution.

## (d) Alexander's Hierarchical Decomposition

In terms of our dichotomy between goal structure models and hierarchical planning models, Alexander's hierarchical decomposition may be viewed as a procedure which combines elements of both domains. The model has four phases:

(1) Formulation of requirements;

(2) Estimation of interactions among requirements;

(3) Decomposition: the result of which, is a "program" for the solution of the problem;

(4) Solution of the problem according to the "program" derived in phase (3).

(*) as in the "missionaries and cannibals" problem: (G. W. Ernst; "GPS and Decision Making: An Overview", in R. Banerji, M. D. Mesarovic, eds., Theoretical Approaches to Non-Numerical Problem Solving, New York, N. Y., Springer-Verlag, 1970), p. 63.

(39) C. Alexander; op. cit.

The list of requirements is not a list of goal variables, but rather a set of variables which specify misfits in the environment. Requirements are further constrained in that they must be as equal as possible in importance, and independent of each other (i.e. each is important to the problem by itself, and not in terms of contributing to another requirement). Relations among requirements specify form implications, not evaluation relations; they measure the "difficulty" in finding solutions which will satisfy any two requirements simultaneously.

In the interaction phase, requirements are taken two at a time, and a binary judgment is made as to whether or not the form implications of one requirement conflict or concur with the form implications of the other: if so, an interaction is present. The results are represented in a matrix of interactions: (cf. Figure No. 3.6

$$\begin{array}{c|c|c|c|c|c|}
 & x_1 & \cdots & x_j & \cdots & x_n \\
\hline
x_1 & c_{11} & \cdots & c_{1j} & \cdots & c_{1n} \\
\hline
\vdots & \vdots & & \vdots & & \vdots \\
\hline
x_i & & \cdots & c_{ij} & \cdots & \\
\hline
\vdots & \vdots & & \vdots & & \vdots \\
\hline
x_n & c_{n1} & \cdots & & \cdots & c_{nn} \\
\hline
\end{array}$$

Figure No. 3.6

where $X = (x_1, x_2, \ldots, x_i, \ldots, x_n)$ is the set of requirements,

$c_{ij} = 0$ if there is no interaction between $x_i$ and $x_j$,

$= 1$, otherwise.

The interactions and requirements are represented as a linear graph, with the requirements as nodes and the interactions as links: (cf. Figure No. 3.7)

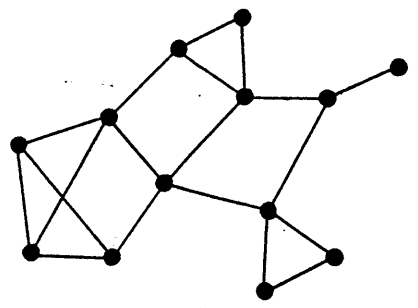


Figure No. 3.7

The decomposition phase successively partitions the graph at points of least information transfer, to a stage where subsets of the graph are small enough as subproblems for the designer to be able to handle them conveniently in a design solution. Figure No. 3.8 illustrates how the linear graph of Figure No. 3.7 might be partitioned and represented as a hierarchical tree or "semi-lattice".

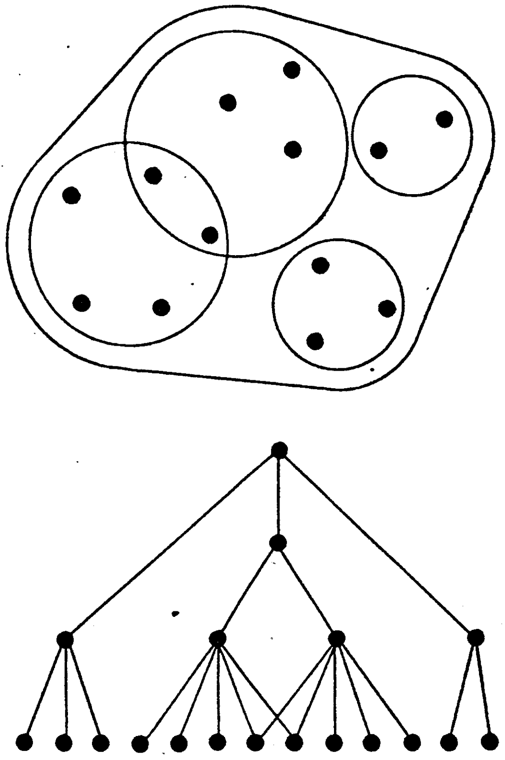
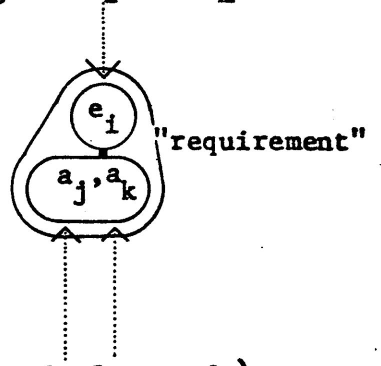


Figure No. 3.8

Diagramming begins at the bottom of the hierarchy, where each of the subsystems is dealt with as a separate design problem. A convention is used for all diagrams such that each diagram contains the essential relational features of its subset, and as little else as possible. The diagrams are combined according to the program indicated by the decomposition tree, until one diagram is completed which shows all the essential features of the design.

The logic of Alexander's model is similar to that of means-ends analysis as a sequential decision process: the problem is decomposed into sub-problems, the sub-problems are solved, and then recombined to yield the solution to the larger problem. From the perspective of our dichotomy between evaluation and search, however, his reasoning is quite different. There is no means in Alexander's model for predicting the performance of alternatives, or for determining their level of achievement with respect to requirements or goals. This results because the designer, in deriving the interactions between requirements, makes prior judgments about certain predicted consequences of the design before he actually generates it. The requirements are intended to "imply" form, but not "specify" it; Alexander wants them to be both "partly open and partly closed" constraints. In effect, the model attempts to link directly small subsets of design attributes and evaluators into "requirements" and anticipate the prediction and evaluation phases of the planning process in determining interactions. Figure No. 3.9 illustrates this point:

evaluators: $G = (e_1, e_2, e_3, \ldots, e_i, \ldots, e_m)$



design attributes:

$A = (a_1, a_2, \ldots, a_j, a_k, \ldots, a_n)$

Figure No. 3.9

The complex functional mapping of attributes onto evaluation space through prediction (cf. Figure No. 3.10):

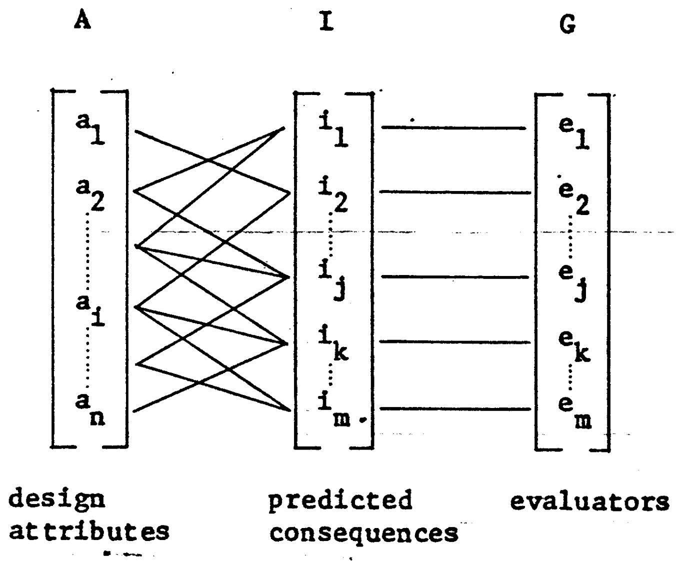


Figure No. 3.10

has been replaced by an ensemble of certain attribute and evaluator sets (in the form of requirements), yet complex interactions among the requirements. (cf. Figure No. 3.11).

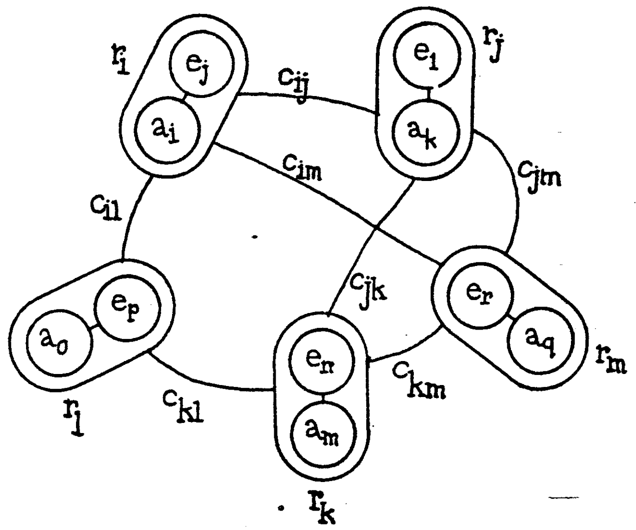


Figure No. 3.11

There are no empirical methods for getting the correlations represented by interactions, apart from previous experience, which cannot be completely valid. Alexander's method then, is more properly, a search strategy, in which these interactions are regarded as prior judgments about crucial issues in the design problem. His attempted integration of evaulation and descriptor languages is provocative, but still requires a posterior evaluation phase in which resulting consequences and worths are assessed, (and the "prior" hierarchical structure of the problem may be revised).

The central conceptual issue appears to lie in the structuring of inherent problem complexity, particularly when uncertainty is introduced. If we accept Alexander's linkage of attributes and evaluators into requirements, a complex-semi-lattice hierarchical decomposition results, which may have to be altered on posterior analysis. On the other hand, if we accept separate, simple goal and action decompositions, problem complexity is transferred to the mapping between these hierarchicies. Our natural tendency is to prefer the latter model, given our concern with evaluation. More importantly, Alexander's model, in not alleviating the need for prediction and evaluation, also does not structure the problem in a form which lends itself to the aggregation of preferences. Thus, it is difficult to compare and assess alternatives, except at the smallest subcomponent "simplex" level. Accordingly, for this paper, we retain the separation of evaluation and descriptor languages; i.e. the distinction between hierarchical goal and planning models.

The remaining basic issue with respect to hierarchical goal models, is whether to accept the "tree" goal decomposition of means-ends analysis, Miller, Fishburn, etc., which also implies computability, or whether to acknowledge the "lattice" decomposition suggested by Manheim and Hall, who permit value-wise dependence among evaluators. Arguments for simplicity in the goal fabric so as to make it easy to compute compound utilities, are valid, but not central to the issue. Fishburn (40) suggests that interdependent goal variables should be aggregated or recast into independent utilities so as to allow the use of additive utilities. However, this is expedient also. The most useful argument for evaluator independence can be derived from Torgerson (41), who distinguishes three kinds of measurement:

(1) Fundamental measurement;

(2) Measurement by arbitrary definition;

(3) Derived measurement.

Even at an elemental level, utilities or preference measures are derived from fundamental attributes of the system, which cannot be inferred directly. Therefore, evaluators which are inter-

(40) P. C. Fishburn; op. cit. (1964), p. 346.

(41) W. S. Torgerson; op. cit., p. 21.

dependent, must be in some sense, derived from a common fundamental measure. Thus, it appears valid to attempt to reformulate evaluators in such a way that each evaluator is derived from a distinct, non-repeated fundamental measure. The formidable difficulties that may be involved in attempting to this operationally, should not be discounted, though.

## 3.2 Hierarchical Planning Models

In Miller, Galanter & Pribram's theory of behaviour, a Plan is:

"...any hierarchical process in the organism that can control the order in which a sequence of operations is to be performed". (42)

Planning is concerned with the strategic aspects of behaviour rather than simply tactical actions: an organism which plans, maintains an internal representation of a complete course of action or "strategy". Hierarchical planning, then, is concerned with "strategies of strategies": the components over which control is maintained, are themselves plans, rather than direct courses of action. The purpose of this "metaplanning" is to control and economize on the organism's search effort.

Miller et. al. also note the interrelationships between values and the execution of plans:

(42) Miller et. al.; op. cit., p. 16.

"...the test phases of the more strategic portions of a Plan are associated with overriding evaluations. Thus a hierarchy of TOTE units may also represent a hierarchy of values." (43)

In hierarchical planning models, the association of plans with values varies from level to level. Lowest level plans may be thought of as points in n-dimensional evaluation space, with utility or value functions associated with each point. Each utility is a composite function of the goal variables which define the point. Higher level plans comprise regions in which a number of lowest-level plans may be nested: since these regions each encompass a number of points, (not all of which have the same utility), high level plans have distributions of values, rather than single values. This notion underlies Manheim's Hierarchical Structure model (44) as well as the Dynamic Evaluation model developed in the thesis.

(a) <u>Hierarchical Structure</u>

Hierarchical Structure is conceived of as a "metaprocess" in that it is concerned with organizing the planning process, rather than specifying solutions to a given problem.(*) The process of finding a solution proceeds by a series of "experiments" or

(43) Ibid.; p. 63.

(44) M. L. Manheim; op. cit. (1966).

(*) Some ways in which the model also gives information about the nature of solutions are described on page operations in which information is acquired about the nature of possible solutions.

The "hierarchical structure" of the design process is the specification of levels of description ("strata" in Mesarovic's terminology (45)) from very general plans, down to solutions specified in all the detail necessary for implementation. Only lowest level designs (i.e. at the most completely specified level of detail) can be considered as solutions to the problem. Levels in the structure are defined in terms of the precision and discriminability among actions in the action space. The concept of "metric" is related to that of level:

> "The metric of an operator (i.e. search-selection procedure) is a division of the action space into sets of actions such that the selection of the SLO (Single Level Operator) can distinguish between two actions if they are from different sets, but cannot distinguish between actions from the same set. A metric is a set of exhaustive disjoint subsets of the set of points in the action space." (46)

Metrics should be chosen so that there is a high degree of difference between actions on the same level, but a high degree of similarity among lower-level actions included within these actions. Alexander's Hierarchical Decomposition algorithm (47) partitions a set of interdependent elemental variables according

(45) Mesarovic et. al.; op. cit., p. 37

(46) Manheim; op. cit., p. 35.

(47) C. Alexander; op. cit.

to roughly the same requirement. Each decomposition level has the same total set of elemental variables, but differs from other levels in the way that the variables are grouped. In this sense, a metric may be viewed as a framework which is appropriate to describe and identify the structuring or grouping at each level.

An experiment is defined as the application of a level operator i to a non-elemental action which was produced previously, to yield another action. (48) The results of an experiment are a new action with its associated cost or value. The new action is at a lower level than the action from which it was produced: as an example, an experiment could be thought of, as the design of large-scale room layouts from smaller-scale and more general floor plans. The cost of executing an experiment depends only on the level, and is constant over the process.

The outcomes of experiments are characterized probabilistically; the action or alternative design resulting from an experiment, and its cost, are uncertain. The model assumes that the analyst has:

> "...a distribution $f_j(\theta)$ for every action j which he has produced so far....Each time he obtains a new action and its associated cost, he acquires information about the true distribution of costs of "actions" included within various non-elemental actions." (49)

$f_j(\theta)$ is the subjective judgment of the likelihood of different values of $\theta$, where $\theta$ is some parameter of costs (i.e. worth) of

(48) M. L. Manheim; op. cit., (1966), p. 43.

(49) Ibid.; p. 46.

experiments. The objective of the model is to determine, at any point in the design process, which "experiment", at what level of detail, should be performed next. The planning process stops when there is no experiment (cost of experiment deducted) which will obtain a significantly better solution than one developed so far.

Each level operator is characterized by a conditional probability distribution which essentially measures the relative amount of information supplied by that operator (i.e. designing at a certain level of detail). This distribution $g_i(y/\theta)$ is defined:

> "Given that some action j is characterized by a particular value of the parameter $\theta$, say $\theta_0$, $g_k(y/\theta_0)$ is the probability that application of operator i to that action will produce an action with a cost equal to y." (50)

Once an experiment is executed, and a design with cost $\hat{y}$ is produced, Bayes theorem is used to revise the analyst's prior distribution, $f'_j(\theta)$:

$$f''_j(\theta/\hat{y}) = \frac{f'_j(\theta)\,g_i(\hat{y}/\theta)}{\sum\limits_{\theta} f'_j(\theta)\,g_i(\hat{y}/\theta)}$$

The prior distributions for actions on higher levels which include the present action j (i.e. its "grandparents" and "parents") are also changed. In these cases, the $g_i(\hat{y}/\theta)$ function remains the same, but the priors for actions at different levels will usually

(50) Ibid.; p. 47.

be different. The prior distributions over ungenerated actions are governed by a homogeneity assumption: their priors are the priors over their parent, least including action, since there is no way of distinguishing them from all others, until they have actually been generated.

The evolution of the planning process is described thus:

> "When the process begins, the marginal distributions $f(\theta_k)$ over the components $\theta_k$ are identical, because no actions have been generated. Each time an experiment is executed, one or more marginals become differentiated. As the process unfolds through the execution of experiments, more and more marginals go off on separate paths in a complex and interrelated manner determined by the inclusion relationships among the actions." (51)

In choosing among possible experiments to do next, the objective is to balance the cost of producing a design alternative against the possible returns: i.e. the hope of getting a less costly or more efficient solution to the problem. For a single stage analysis, the expected utility of each possible experiment is computed by taking each possible result y, computing $p_{ij}(y)$, temporarily updating the status of the process to compute $u(e_{ij},y)$ and integrating over all results y:

$$u(e_{ij}) = \int p_{ij}(y)u(e_{ij},y)\,dy$$

(51) Ibid.; p. 74.

The expected value of terminating (the null experiment) is the value of the best elemental (lowest-level) action so far. The experiment with the highest expected utility is chosen.

For a multiple stage analysis, computation is much more involved, since it must extend out over many linked experiments rather than just one: all possible combinations of 1st stage, 2nd stage,...nth stage experiments must be examined.

In order to reduce computation, several kinds of constraints on the process are suggested:

(1) Sequence constraint: the process must continue through an orderly progression of levels;

(2) Jump-back constraint: once control has been transferred from one level of analysis to a lower level, new actions can no longer be generated at higher levels (this appears to be an unreasonable restriction);

(3) Bandwidth constraint: only a limited number of potential experiments will be examined;

(4) Look-ahead constraint: restricts the number of levels ahead that a new design may be generated from a present action.

Some variations on these constraints are used in the Dynamic Evaluation model.

It should be noted that there are essentially two arbitrary points in the process:

(1) the specification of the initial subjective distribution $f'_j(\theta)$ over the universal design. Given the learning aspect of the model, this is not a serious point: all that is required is that the distribution cover a range of all values of $\theta$ that are likely to be encountered in evaluating lower-level designs.

(2) the specification of the conditional operator characteristics; $g_i(y/\theta)$. An extension of the Hierarchical Structure model mentioned by Manheim (52) and Pecknold (53) allows the analyst to specify a family of distributions for the g function. Through the process, he learns not only about his changes of success (the $f''(\theta/y)$ distributions), but also about the amount of information contained in a Single Level Operator. Denoting $P'(\beta)$ as the estimate of the relative likelihood of different combinations of probability distributions for the operators, the Bayesian model also revises this distribution posteriorly, after observing the result $\hat{y}$, of an experiment:

(52) Ibid.; pp. 163-164.

(53) W. M. Pecknold; The Evolution of Transport Systems: An Analysis of Time-Staged Investment Strategies Under Uncertainty (Cambridge, Mass., unpublished Ph.D. thesis, M.I.T. Dept. of Civil Engineering, 1970), Appendix D.

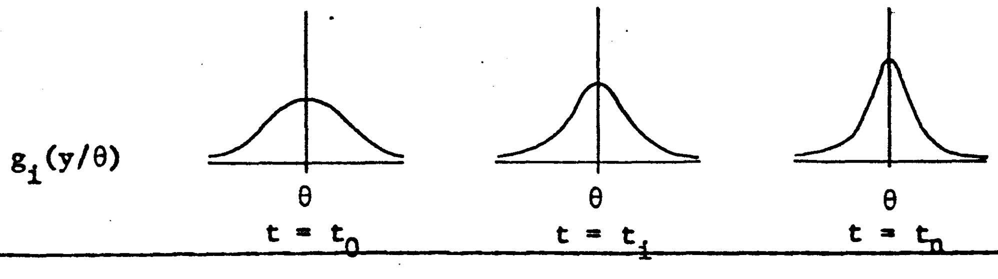
$$P''(\beta/\hat{y}) = \frac{P'(\beta)g_{\beta i}(\hat{y}/\beta)}{\sum\limits_{\beta} P'(\beta)g_{\beta i}(\hat{y}/\beta)}$$

The complete posterior distribution for an action j, is:

$$P_j''(\theta/\hat{y}) = \frac{P_j'(\theta)\sum\limits_{\theta} g(\hat{y}/\beta,\theta)P'(\beta)}{\sum\limits_{\theta} P_j'(\theta)\sum\limits_{\beta} P'(\beta)g(\hat{y}/\beta,\theta)}$$

Pecknold also gives a more exact treatment of this, when it cannot be assumed that the posterior distributions, P"(β) and P"(θ) are independent, and therefore, joint distributions over both β and θ must be used. (54)

Over the history of the process therefore, we would expect the designer to be able to generate intermediate level designs whose expected evaluations are closer to the ultimate distributions of evaluations over their included designs, as yet ungenerated. Thus, the $g_i(y/\theta)$ distribution for a particular level would become "tighter" (i.e. with a smaller coefficient of variation) as the process evolves; for example:



(54) Ibid.; p. D-10.

Further issues such as:

(1) constant costs over level operators;

(2) the function $g_i(y/\theta)$ is constant for all $\theta$: i.e. $g_i(y/\theta)$ can also be expressed as a function $h_i(y - \theta)$;

are discussed under the Dynamic Evaluation model, where they have been simplified or modified, along with other assumptions from Hierarchical Structure.

## 4. COMPUTER AIDED EVALUATION SYSTEM

The following set of programs provides an interactive capability in DISCOURSE for the multi-dimensional evaluation of design or planning projects, whose predicted consequences have been arrayed in an impact matrix. The component routines divide roughly into three areas:

(1) User Operations: a set of independent programs for operating on the impact matrix at any stage in the planning process, with or without an associated preference structure. They provide a variety of means for assessing the current status of the process, for comparing, ranking, checking for dominance, or satisfaction of alternatives with respect to defined goals.

(2) Static Evaluation: terminal assessment of a set of design alternatives with respect to a multi-dimensional preference structure. The program checks for dominance among alternatives and allows the user to select from a number of evaluation methods which assess relative value, value (certainty), or utility (uncertainty) for each alternative.

(3) Dynamic Evaluation: a hierarchically-structured planning model in which every design alternative is evaluated as soon as it has been generated. The evaluations are used to structure the design process, and through Bayesian

revision of prior evaluations, use the experience accumulated by the decision-maker as a guide to future action.

All three groups of programs accept design alternatives which have been structured hierarchically at different levels of generality, but this is a necessary requirement only for the Dynamic Evaluation model. However, if design alternatives have been so structured, then a corresponding goal structure must also be input.

The programs provide for user interaction in accepting descriptions of design alternatives and preference structures, and allowing him to specify at certain points how he wishes execution to proceed. Provided the necessary project information is arrayed in files accessible to DISCOURSE, the programs are self-contained, and may be used without modification. However, a user familiar with DISCOURSE will be able to alter the programs or intervene during execution so as to tailor them more closely to his particular project requirements.

## 4.1 User Operations

The need for independent user evaluation operations can arise during the course of the planning process, whenever the decision-maker wishes to improve his understanding of the problem issues, or the present status of alternatives; without undergoing a full terminal evaluation of all alternatives, as in the "Static" model.

He may also be unwilling to put the planning process under the degree of formal control suggested in the "Dynamic Evaluation" model. Although such procedures are more ad-hoc and less formalized than the other two models to be described, they can contribute in the role of guiding the search process, and preparing the conditions for more complete formal analyses.

User operations comprise a variety of computational procedures for operating on, and manipulating, the impact matrix current at any point in the process. For most operations, limited information about the decision-maker's preference structure is required, since this may not be clarified at intermediate stages. Possible explorations of problem domains include the following:

(1) Value System: varying the preference structure, or trade-offs among evaluators, to judge the impact on the ranking of the present set of alternatives; and in the opposite vein: checking the present performance alternatives with respect to evaluators, and identifying which goal variables or aspiration levels should be adjusted (i.e. the effect of the current levels of achievement in determining the value system).

(2) Solution Space: identifying the significant differences among alternatives; identifying which decision variables to manipulate in the search for improved solutions.

(3) Display: representing the current status of the project: preference structure, solution space, or the mappings between them.

The set of operations currently implemented provides only rudimentary capabilities. Possible extensions to the routines are outlined after the description of available commands.

All user-operation programs require that the following updated project information to be known and resident in the system:

(1) the names of evaluators and designs

(2) the total number of designs (all levels)

(3) the total number of evaluators or goal variables (all levels)

(4) the goal and design structures (if the problem is hierarchically structured)

(5) the overall impact matrix, by evaluators and designs.

If the problem is hierarchically structured in levels, the user selects the level for which he wishes the analysis to be done. The retrieval of this information from disk files and from the user, is carried out by the DISCOURSE program "Preliminary". The flow of control for the programs is shown in Figure No. 4.1.

## USER OPERATIONS

PROGRAM FLOW

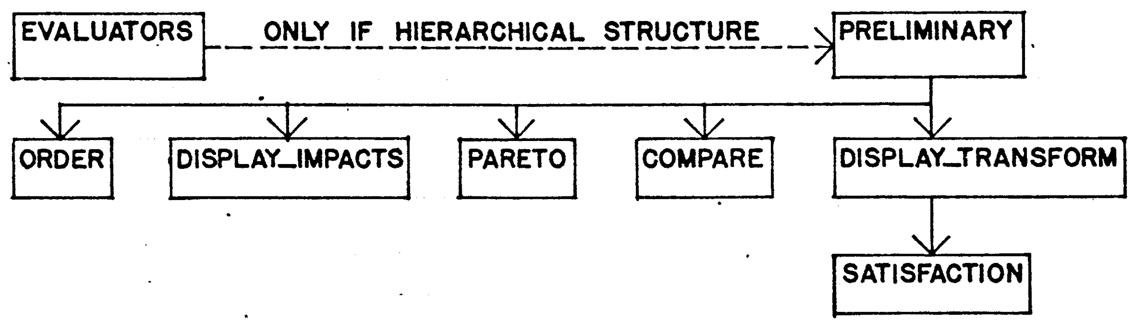


FIGURE NO. 4·1

(a) Available commands:

(1) "order" (all designs on one level, with respect to a named evaluator, by increasing or decreasing values)

The program accepts from the user, the name of the evaluator he has selected, and whether the ranking is to be done by increasing value (highest value receives a rank of 1), or by decreasing value (lowest value receives a rank of 1). "order" derives and displays this ranking in the following format:

| Designs | Ranking |
|---|---|
| name1 | r1 |
| name2 | r2 |
| ⋮ | ⋮ |
| $\text{namen}_1$ | $r_{n1}$ |

(2) "Pareto" (quasi-orders all designs on one level, with respect to all evaluators)

The program first queries the user as to the direction of his preferences for each eavluator. An ordinal ranking matrix for all designs with respect to each evaluator, is derived, and a dominance check is performed by constructing quasi-levels. The PL/1 function "quasi-order" first constructs a "reachability" matrix for all the designs for which each design is better on, ("reaches") for at least one evaluator; quasi-levels are formed by grouping all designs with the same row sum in the reachability matrix. These sums are then ordered; all designs with rank 1 are undominated, and form the Pareto-efficient frontier; designs with ranks 2, 3, etc. form lower quasi-levels, which contain dominated alternatives. The results are displayed as follows:

```
Quasi-level 1:     Pareto-efficient frontier

name1
name2
  ⋮
name i

Quasi-level 2:     Dominated alternatives

namej
namek
  ⋮
Quasi-level n:     Dominated alternatives

namep
nameq
  ⋮
```

(A later version might allow consideration of probabilistic dominance: quasi-levels would then be derived according to some specified criterion of dominance.)

(3) "<u>display-impacts</u>" (for 2 to 5 specified designs, with respect to all evaluators, on one level)

The program accepts from the user, the names of 2 to 5 designs to be displayed; retrives the relevant impacts from the overall impact matrix, and arrays them in the following format:

$$\begin{array}{llll}
\text{Designs:} & \text{name1} & \text{name2} & \cdots\cdots \\
\text{Impacts:} & & & \\
\text{goal } 1 & i_{1,1} & i_{1,2} & \\
\text{goal } 2 & i_{2,1} & i_{2,2} & \\
\vdots & \vdots & \vdots & \\
\text{goal } m_1 & i_{m1,1} & i_{m1,2} &
\end{array}$$

(4) "<u>display-transforms</u>" (for 2 to 5 specified designs, with respect to all evaluators, on one level)

The program accepts from the user, the names of 2 to 5 designs to be displayed. A step value function array for each evaluator is presupposed. (A later version might allow the functional form of the utility curve to be input as well.) It then maps the reduced impact matrix onto the value array, and displays the resulting value transformation in the following format:

$$\begin{array}{llll}
\text{Designs:} & \text{name1} & \text{name2} & \cdots \\
\text{Transforms:} & & & \\
\text{goal } 1 & t_{1,1} & t_{1,2} & \\
\text{goal } 2 & t_{2,1} & t_{2,2} & \\
\vdots & \vdots & \vdots & \\
\text{goal } m_1 & t_{m1,1}. & t_{m1,2} &
\end{array}$$

(5) "<u>compare</u>" (2 selected designs with respect to all evaluators, on one level)

The program accepts from the user, the names of 2 designs to be compared. For each evaluator, the difference between impact 1 and impact 2 is computed, and displayed, along with the present impacts for both designs.

| | name1 | name2 | Difference 1 - 2 |
|---|---|---|---|
| Evaluators | | | |
| goal 1 | $i_{1,1}$ | $i_{1,2}$ | $(\pm)d_1$ |
| goal 2 | $i_{2,1}$ | $i_{2,2}$ | $(\pm)d_2$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| goal $m_1$ | $i_{m1,1}$ | $i_{m1,2}$ | $(\pm)d_{m1}$ |

(6) "<u>satisfaction</u>" (for 1 selected design, with respect to all evaluators, on one level)

The program accepts from the user, the name of the design to be analysed. A value transform array is presupposed (or program "display-transform" has already been executed). The utility or transform values for each evaluator are ranked and displayed in the following format:

| Evaluator | Utility | Rank | |
|---|---|---|---|
| goal 1 | $t_{1,k}$ | $r_1$ | |
| goal 2 | $t_{2,k}$ | $r_2$ | best |
| ⋮ | ⋮ | ⋮ | worst |
| goal $m_1$ | $t_{m1,k}$ | $r_{m1}$ | |

With "satisfaction", it is also of interest to compute utility satisfaction ranks for a number of selected designs, and call the PL/1 function "quasi-order". The resulting quasi-levels will give the evaluators which are consistently well-satisfied among all designs (Quasi-level 1), to those which are consistently poorly satisfied among all designs (lower quasi-levels).

(b) Possible Extensions

Useful additions to the current repertoire of available commands would provide for more sophisticated identification of problem issues. Two areas suggest themselves:

(1) Break-even analysis

(2) Incremental or marginal improvement.

Computationally, some of these routines are quite complex, since they require more intervention in the planning process than present commands, approaching the level of evaluation models.

Break-even analysis: Deals with the question: "what tradeoff ratios among evaluators would produce indifference among all alternatives at one level?" For the linear scoring function; a solution may be attempted by simultaneous linear equations;

$$\begin{array}{cccccccc}
\lambda_1 e_{11} & + & \lambda_2 e_{21} & + & \cdots & + & \lambda_m e_{m1} & = k, \\
\lambda_1 e_{12} & + & \lambda_2 e_{22} & + & \cdots & + & \lambda_m e_{m2} & = k, \\
\vdots & & \vdots & & \cdots & & \vdots & \vdots \\
\lambda_1 e_{1n} & + & \lambda_2 e_{2n} & + & \cdots & + & \lambda_m e_{mn} & = k.
\end{array}$$

Since the resulting form is n equations in m unknowns, if $n = m$, then a solution is possible. However, if $n > m$, then the set may be overconstrained and insoluble; if $n < m$, the set is underspecified. In this latter case though, tradeoff values may be expressed in terms of one another, and a variety of solutions under constraints, tried.

Simpler forms of break-even analysis could address the following questions for any two alternatives at one level:

(a) for any one evaluator, what change in its tradeoff coefficient would make the two alternatives indifferent to one another, if at all?

(b) for any one evaluator, what changes in the impact value of one alternative would be required to make two alternatives equal or indifferent, if at all? Since this involves mapping of impact values onto the preference structure, a number of solutions may have to be attempted before approximating indifference. For quasi-separable utilities, the added computational cost of aggregating for all levels, must be considered.

(c) for a particular set of design attributes (i.e. control variables) which can be varied incrementally, what changes in one design are required to get indifference between twl alternatives? Since design attributes must pass through two complex mappings, predicting the direction of value shifts for any one attribute change, may be quite difficult. (the added computational cost for quasi-separable utility aggregation must also be considered.) Incorporating a Bayesian learning model within the routine would aid in the prediction of value shifts.

Incremental improvement is simpler than break-even analysis, but deals with similar issues: the evaluation effects of incremental variation in certain control variables:

(a) for a particular predicted impact for one design: change the impact incrementally by a significant amount, and determine the effect, if any, on the overall ranking of alternatives.

(b) for any one evaluator $e_i$ for all designs, change incrementally the weight $\lambda_i$ assigned to $e_i$ by a significant amount (normalizing the other weights $(\lambda_1, \ldots, \lambda_{i-1}), (\lambda_{i+1}, \ldots \lambda_m)$ in the process), and determine the effect, if any, on the overall ranking of alternatives. This is not applicable to the Fishburn or Case relative value methods, where the weighting is proportional to the spread of evaluator values among alternatives.

(c) for any one or more control variables of a design: change incrementally the variables by significant amounts, and determine the effect if any, on the overall ranking of alternatives.

As with (c) above, this is a complex search issue, which probably must be integrated within a learning model in order to improve efficiency.

## 4.2 Static Evaluation

"Static Evaluation" refers to evaluation carried out in the context of the standard statistical decision problem. The operation is performed near the end of the design process; a large number of alternatives (which may or may not be developed to several levels of detail) is assessed at one pass (hence the term "static"). As with user operations, a good deal of preparatory information specific to the project under consideration, is required before evaluation can take place.

The DISCOURSE programs "Preliminary" (Figure No. 4.2) and "Evaluators" (only if applicable) retrieve the initial project information required:

(1) the names of evaluators and designs;

(2) the total number of designs (all levels);

(3) the total number of evaluators or goal variables (all levels);

(4) the goal and design structures (if the problem is hierarchically structured);

(5) the overall impact matrix.

The user selects the level for analysis from the console.

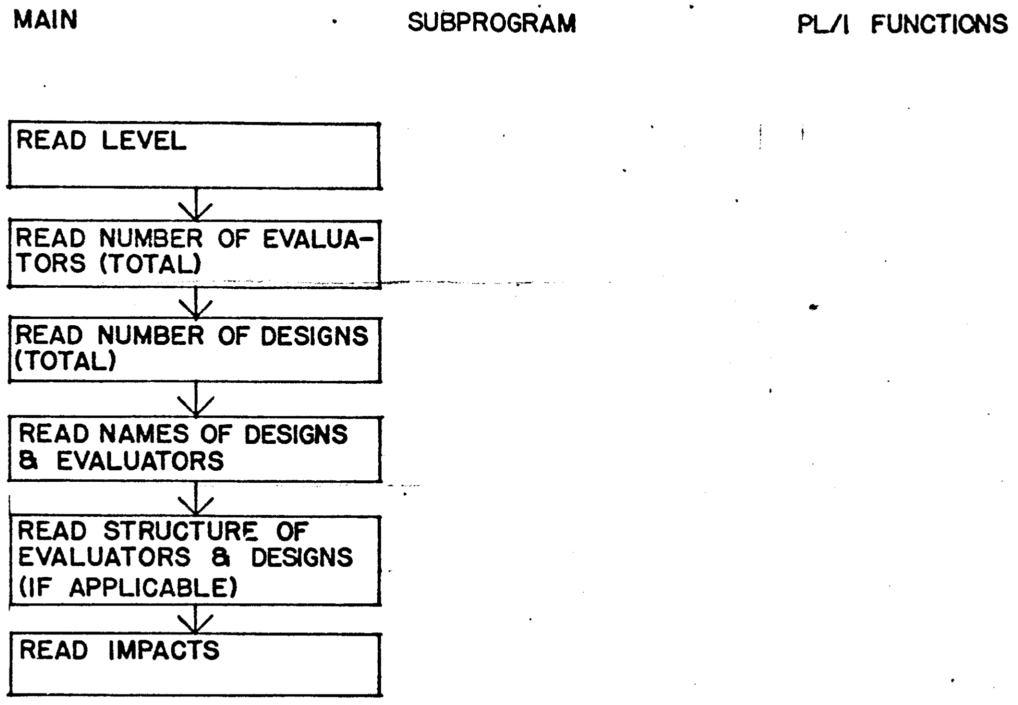

MAIN
SUBPROGRAM
PL/I FUNCTIONS
READ LEVEL
READ NUMBER OF EVALUATORS (TOTAL)
READ NUMBER OF DESIGNS (TOTAL)
READ NAMES OF DESIGNS & EVALUATORS
READ STRUCTURE OF EVALUATORS & DESIGNS (IF APPLICABLE)
READ IMPACTS

FIGURE NO. 4·2

Figure No. 4.3 represents the Process Flow Chart and the associated Project-Dependent Information retrieved from disk files, and updated by the output from subroutines. The overall Program Control is illustrated in Figure No. 4.4. Once an overall Prediction Phase has been completed, (which serves to transform design descriptor attributes into evaluation attributes), the DISCOURSE program "Single-Pass" (Figure No. 4.5) is called, which:

(1) derives rankings for all designs with respect to each evaluator (the Ordinal Ranking Matrix). Alternatively, an Ordinal Impact Matrix may serve as input to the program (but then, the analysis cannot be carried beyond step 2).

(2) checks for dominated alternatives by constructing quasi-levels. If dominated alternatives occur, they are deleted, and the impact matrix is reduced accordingly.

(3) queries the user as to which of several evaluation methods he wants to use for assessing the remaining un-dominated alternatives. The requisite value or utility functions and weights are assumed to be available before this choice is made. (Alternatively, the process control may transfer out of the computer environment so as to allow the user to prepare the necessary preference functions, input them, and transfer back to the final computer-based assessment.)

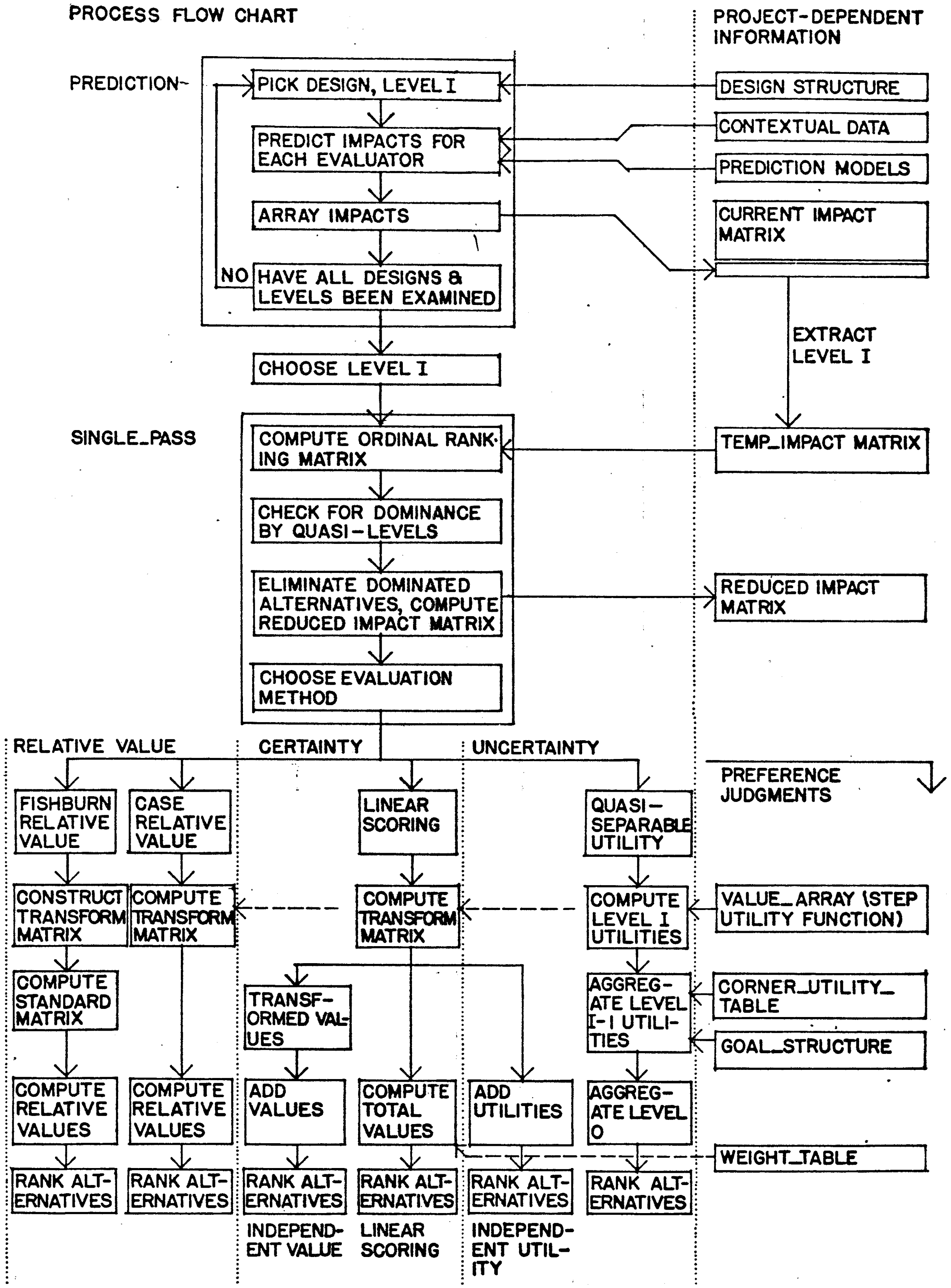

PROCESS FLOW CHART
PROJECT-DEPENDENT INFORMATION
PREDICTION
PICK DESIGN, LEVEL I
PREDICT IMPACTS FOR EACH EVALUATOR
ARRAY IMPACTS
NO
HAVE ALL DESIGNS & LEVELS BEEN EXAMINED
DESIGN STRUCTURE
CONTEXTUAL DATA
PREDICTION MODELS
CURRENT IMPACT MATRIX
EXTRACT LEVEL I
CHOOSE LEVEL I
SINGLE_PASS
COMPUTE ORDINAL RANKING MATRIX
TEMP_IMPACT MATRIX
CHECK FOR DOMINANCE BY QUASI-LEVELS
ELIMINATE DOMINATED ALTERNATIVES, COMPUTE REDUCED IMPACT MATRIX
REDUCED IMPACT MATRIX
CHOOSE EVALUATION METHOD
RELATIVE VALUE
CERTAINTY
UNCERTAINTY
PREFERENCE JUDGMENTS
FISHBURN RELATIVE VALUE
CASE RELATIVE VALUE
LINEAR SCORING
QUASI-SEPARABLE UTILITY
CONSTRUCT TRANSFORM MATRIX
COMPUTE TRANSFORM MATRIX
COMPUTE TRANSFORM MATRIX
COMPUTE LEVEL I UTILITIES
VALUE_ARRAY (STEP UTILITY FUNCTION)
COMPUTE STANDARD MATRIX
TRANSFORMED VALUES
AGGREGATE LEVEL I-1 UTILITIES
CORNER_UTILITY_TABLE
GOAL_STRUCTURE
COMPUTE RELATIVE VALUES
COMPUTE RELATIVE VALUES
ADD VALUES
COMPUTE TOTAL VALUES
ADD UTILITIES
AGGREGATE LEVEL O
WEIGHT_TABLE
RANK ALTERNATIVES
RANK ALTERNATIVES
RANK ALTERNATIVES
RANK ALTERNATIVES
RANK ALTERNATIVES
RANK ALTERNATIVES
INDEPENDENT VALUE
LINEAR SCORING
INDEPENDENT UTILITY

FIGURE NO. 4·3

PROGRAM CONTROL

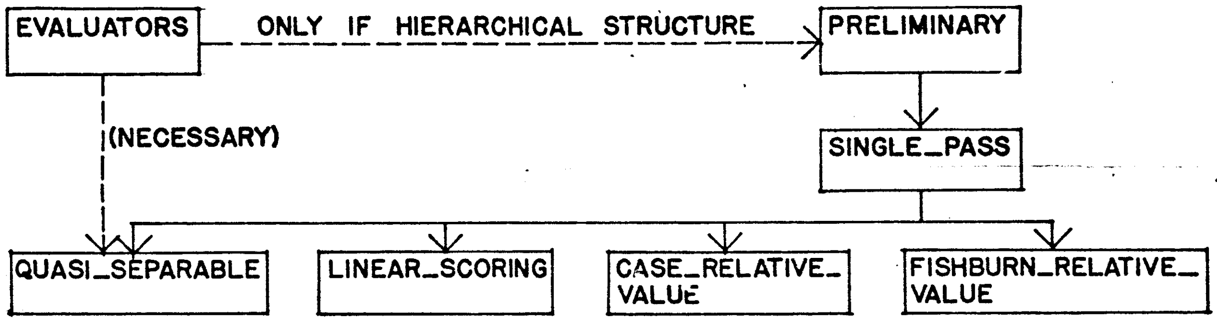


FIGURE NO. 4·4

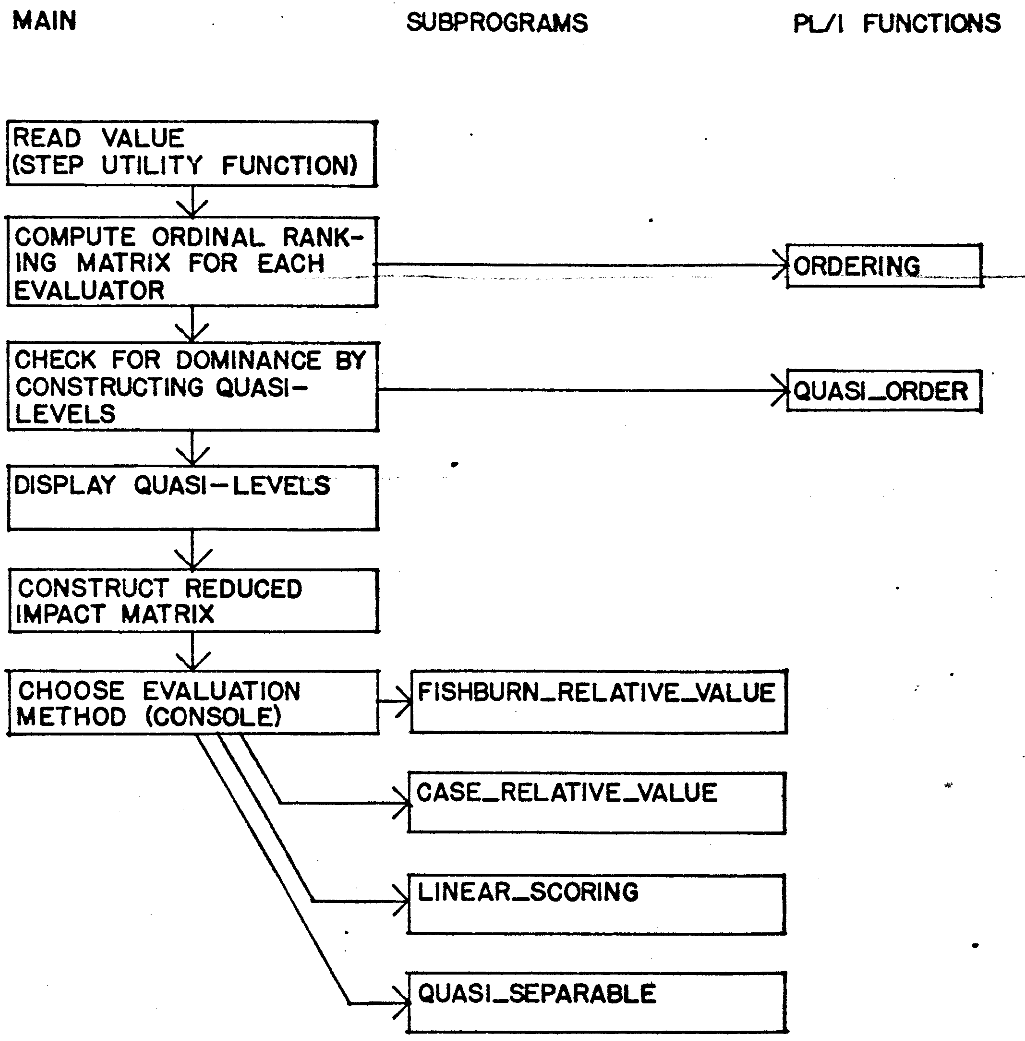

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
READ VALUE (STEP UTILITY FUNCTION)
COMPUTE ORDINAL RANKING MATRIX FOR EACH EVALUATOR
ORDERING
CHECK FOR DOMINANCE BY CONSTRUCTING QUASI-LEVELS
QUASI_ORDER
DISPLAY QUASI-LEVELS
CONSTRUCT REDUCED IMPACT MATRIX
CHOOSE EVALUATION METHOD (CONSOLE)
FISHBURN_RELATIVE_VALUE
CASE_RELATIVE_VALUE
LINEAR_SCORING
QUASI_SEPARABLE

FIGURE NO. 4·5

Four evaluation methods are available, which cover a variety of possible value systems and decision environments:

(a) Fishburn Relative Value (relative value)

(b) Case Relative Value (relative value)

(c) Linear Scoring Function (certainty and uncertainty)

(d) Quasi-additive utility functions (uncertainty).

In the present implementation, no distinction is made between uncertainty and certainty: the utility values derived are assumed to represent "expected utility". Adding a capability for assessment of probabilistically distributed impacts, is simple conceptually, but increases the size of the impact matrix by a factor of (2 x the number of probability steps) and the number of utility computations similarly. Also, the storing of utilities and values as step functions, makes no distinction as to how the original preference structure was derived: through indifference curve analysis, analytical function solution, canonical lottery results, etc. The ability to assess and input user preferences directly, could also be added to the system.

From the point of view of the decision-maker, the distinction between "relative value" and "utility" is important only if the Static Evaluation is not going to be truly "static" (i.e. with a fixed preference structure incrementally, etc. Both "relative value" and "utility" are "relative" in the sense of applying to a single decision-maker: utility is commonly held to be not inter-personally comparable among decision-makers. However, in our distinction, "relative value" is also held to be relative to the present set of alternatives and their associated impact values: adding another non-dominated alternative to the set of designs under consideration (or deleting one from the set), requires that the relative values for all alternatives and evaluators be re-calculated.

On the other hand, "utility" is held to be portable in the sense that another alternative may be included for evaluation, and yet, the current worth of the present alternatives will not change (although the rankings among them, being an ordinal and therefore, relative measure, will change). This is assured by assessing utilities over a large number of consequences and "pseudo-consequences" for each evaluator; pseudo-consequences being values of the goal variable that future alternatives might take on. Once an alternative is gneerated, a set of real consequence values is selected from the set of possible consequences and pseudo-consequences.

Since assessments resulting from the use of different evaluation methods are not strictly comparable, the decision-maker must clarify which interpretations he wishes to be put on the "worth" of an alternative, before selecting the appropriate procedure.

(a) Fishburn Relative Value

Fishburn's general additive value model (55) is similar to the Linear Scoring Function in terms of requiring independence among evaluators, but is more rigorous in its determination of the relative importance of different parameters. Figure No. 4.6 outlines the logic of the DISCOURSE program, "Fishburn-Relative-Value":

(1) The impact matrix of design consequences is mapped onto the value array or function, for each evaluator, to yield a transformed matrix:

Impact Matrix — Value Array — Transform Matrix

$$
\begin{array}{c}
\begin{array}{c|ccc|}
 & A_1 \cdots & A_j \cdots & A_n \\
\hline
E_1 & i_{11} & \cdots & i_{1n} \\
\vdots & \vdots & & \vdots \\
E_i & i_{i1} & \bigcirc\!\!\!\!\!\; i_{ij} & \vdots \\
\vdots & \vdots & & \vdots \\
E_m & i_{m1} & \cdots & i_{mn} \\
\hline
\end{array} \\
(i_{ij} = k)
\end{array}
\longrightarrow
\begin{array}{c|ccc|}
 & 1 \cdots & k \cdots & r \\
\hline
E_1 & v_{11} \cdots & v_{1k} \cdots & v_{1r} \\
\vdots & & & \vdots \\
E_i & & v_{ik} & \vdots \\
\vdots & & & \\
E_m & & & v_{mr} \\
\hline
\end{array}
\longrightarrow
\begin{array}{c|ccc|}
 & A_1 \cdots & A_j \cdots & A_n \\
\hline
E_1 & t_{11} \cdots & t_{1j} \cdots & t_{1n} \\
\vdots & \vdots & & \vdots \\
E_i & \vdots & t_{ij} & \vdots \\
\vdots & \vdots & & \vdots \\
E_m & t_{m1} & \cdots & t_{mn} \\
\hline
\end{array}
$$

(2) A standardized score matrix is constructed from the transform matrix by stepping through each evaluator, assigning the consequence with the best transform

(55) P. C. Fishburn; op. cit.; (1965)

FISHBURN_RELATIVE_VALUE (DISCOURSE)

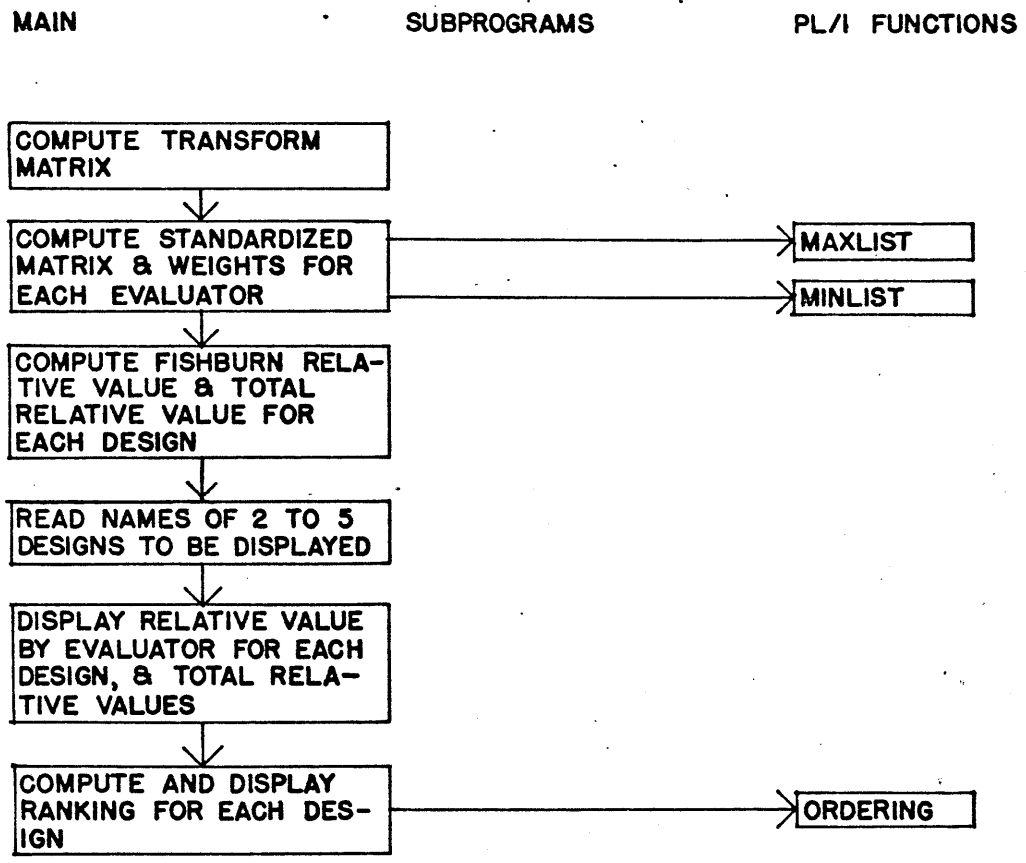

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
COMPUTE TRANSFORM MATRIX
COMPUTE STANDARDIZED MATRIX & WEIGHTS FOR EACH EVALUATOR
MAXLIST
MINLIST
COMPUTE FISHBURN RELATIVE VALUE & TOTAL RELATIVE VALUE FOR EACH DESIGN
READ NAMES OF 2 TO 5 DESIGNS TO BE DISPLAYED
DISPLAY RELATIVE VALUE BY EVALUATOR FOR EACH DESIGN, & TOTAL RELATIVE VALUES
COMPUTE AND DISPLAY RANKING FOR EACH DESIGN
ORDERING

FIGURE NO. 4·6

value, 1.0; the worst 0.0; and linearly scaling the other transform values between these two bounds. Denoting $t_i^*$ as the value of the best consequence, for evaluator i; and $t_{i*}$ as the value of the worst consequence, for evaluator i; we have, for any element $s_{ij}$ of the standardized matrix:

$$s_{ij} = \frac{t_{ij} - t_{i*}}{t_i^* - t_{i*}}$$

Transform Matrix

$$\begin{array}{cc} & \begin{array}{ccc} A_1 \cdots & A_j \cdots & A_n \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{array} & \begin{bmatrix} t_{11} & \cdots t_{1j} \cdots & t_{1n} \\ \vdots & & \vdots \\ \vdots & \enclose{circle}{t_{ij}} & \vdots \\ \vdots & & \vdots \\ t_{m1} & \cdots & t_{mn} \end{bmatrix} \end{array}$$

Standardized Matrix

$$\begin{array}{cc} & \begin{array}{ccc} A_1 \cdots & A_j \cdots & A_n \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{array} & \begin{bmatrix} s_{11} & \cdots s_{1j} \cdots & s_{1n} \\ \vdots & & \vdots \\ \vdots & \enclose{circle}{s_{ij}} & \vdots \\ \vdots & & \vdots \\ S_{m1} & \cdots & s_{mn} \end{bmatrix} \end{array}$$

(3) The relative weight for each evaluator i, is determined by the difference between its highest and lowest transform values:

$w_i = (t_i^* - t_{i*})$  $rv_{ij} = w_i s_{ij}$, where

$rv_{ij}$ is the relative value of alternative j with respect to evaluator i.

(*) note that this is roughly equivalent to the conjoint scaling procedure mentioned on page 38.

(4) The total relative value is derived by multiplying each $s_{ij}$ by its appropriate weight $w_i$, and summing over all evaluators, for each design:

$$TRV_j = \sum_{i=1}^{m} w_i s_{ij}$$

(5) All designs are ranked by total relative value.

(6) The program accepts from the user, the names of 2 to 5 designs to be displayed. The computational results: relative values for each design with respect to each evaluator, total relative value for each design, and ranking for each design, are displayed:

| Designs: | name 1 | name 2 |
|---|---|---|
| Relative Values | | |
| goal 1 | $rv_{11}$ | $rv_{12}$ |
| goal 2 | $rv_{21}$ | $rv_{22}$ |
| goal m | $rv_{m1}$ | $rv_{m2}$ |
| Total: | $TRV_1$ | $TRV_2$ |
| Ranking: | $r_1$ | $r_2$ |

(b) Case Measure of Relative Value

The Case measure of relative value (56) assumes that one can obtain from the decision-maker, a set of probabilities $(p_1,\ldots,p_n)$ such that the consequences for all alternatives, when multiplied by their respective probabilities, are equally preferred; i.e.:

$p_{i1}i_{i1} \sim p_{i2}i_{i2} \sim p_{i3}i_{i3} \sim \ldots\ldots \sim p_{in}i_{in}$, for each evaluator i.

In each case where $p_{ij}$ refers to the probability of obtaining consequence $i_{ij}$, the alternative outcome with probability $(1 - p_{ij})$ is assumed to be the "status quo". On the assumption of maximizing expected value, we derive:

$p_{i1}\cdot rv_{i1} = p_{i2}\cdot rv_{i2} = p_{i3}\cdot rv_{i3} = \ldots = p_{in}\cdot rv_{in}$, for each evaluator i

where $rv_{ij}$ is the relative value of alternative j for evaluator i.

In the program "Case-Relative-Value" (Figure No. 4.7), the procedure:

(1) maps the impact matrix onto a value function or array

(56) R. L. Ackoff; Scientific Method: Optimizing Applied Research Decisions (New York, N. Y., John Wiley & Sons, Inc.; 1962), pp. 91-93.

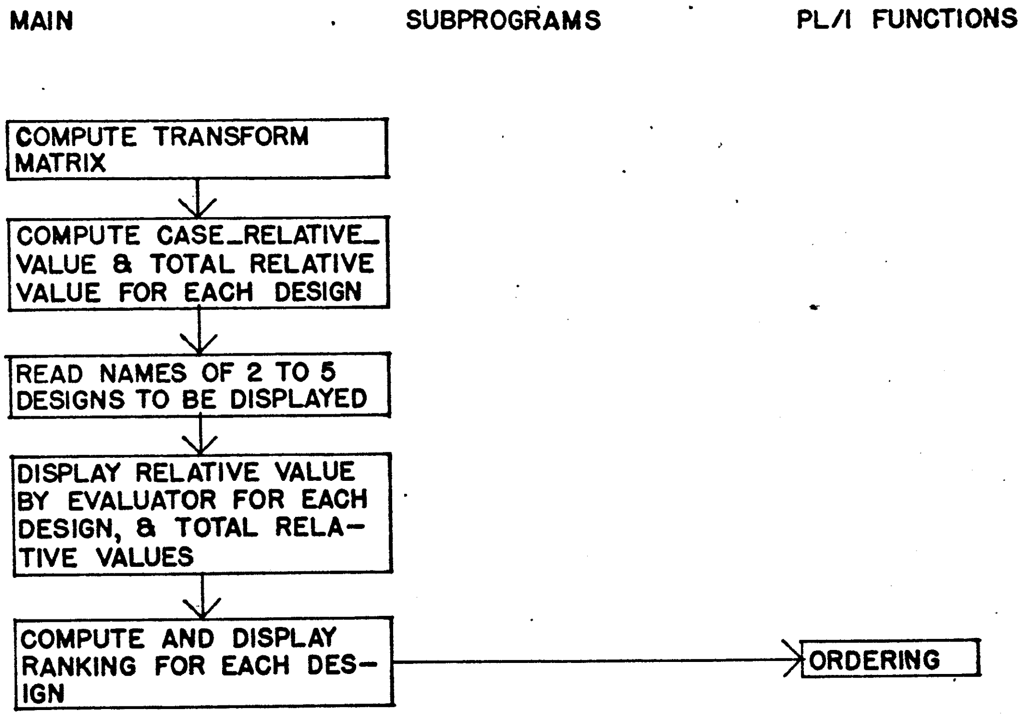

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
COMPUTE TRANSFORM MATRIX
COMPUTE CASE_RELATIVE_VALUE & TOTAL RELATIVE VALUE FOR EACH DESIGN
READ NAMES OF 2 TO 5 DESIGNS TO BE DISPLAYED
DISPLAY RELATIVE VALUE BY EVALUATOR FOR EACH DESIGN, & TOTAL RELATIVE VALUES
COMPUTE AND DISPLAY RANKING FOR EACH DESIGN
ORDERING

FIGURE NO. 4·7

for each evaluator, to yield the probabilities matrix:

<u>Impact Matrix</u> <u>Value Array</u> <u>Probabilities Matrix</u>

$$\begin{array}{cc} & \begin{array}{ccccc} A_1 & \cdots & A_j & \cdots & A_n \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{array} & \left[\begin{array}{ccccc} i_{11} & \cdots & i_{1j} & \cdots & i_{1n} \\ \vdots & & & & \vdots \\ \vdots & & \textcircled{$i_{ij}$} & & \vdots \\ \vdots & & & & \vdots \\ i_{m1} & \cdots & \cdots & \cdots & i_{mn} \end{array}\right] \end{array} \longrightarrow \begin{array}{cc} & \begin{array}{ccccc} 1 & \cdots & k & \cdots & r \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{array} & \left[\begin{array}{ccccc} v_{11} & \cdots & v_{1k} & \cdots & v_{1r} \\ \vdots & & & & \vdots \\ \vdots & & \textcircled{$v_{ik}$} & & \vdots \\ \vdots & & & & \vdots \\ v_{m1} & \cdots & \cdots & \cdots & v_{mr} \end{array}\right] \end{array} \longrightarrow \begin{array}{cc} & \begin{array}{ccccc} A_1 & \cdots & A_j & \cdots & A_n \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{array} & \left[\begin{array}{ccccc} p_{11} & \cdots & p_{1j} & \cdots & p_{1n} \\ \vdots & & & & \vdots \\ \vdots & & \textcircled{$p_{ij}$} & & \vdots \\ \vdots & & & & \vdots \\ p_{m1} & \cdots & \cdots & \cdots & p_{mn} \end{array}\right] \end{array}$$

$(i_{ij} = k)$

Alternatively, the program can accept direct input of the probabilities matrix, since operationally, its direct assessment would require fewer judgments from the decision-maker than the derivation of a value function for each evaluator.

(2) letting $\sum_j rv_{ij} = 1.0$ (or any arbitrarily selected constant) obtains the value of $rv_{i1}$ from:

$$rv_{i1} + \frac{p_{i1} rv_{i1}}{p_{i2}} + \frac{p_{i1} rv_{i1}}{p_{i3}} + \ldots + \frac{p_{i1} rv_{i1}}{p_{in}} = 1.0$$

$$rv_{i1} = \frac{1}{(1 + \frac{p_{i1}}{p_{i2}} + \frac{p_{i1}}{p_{i3}} + \ldots + \frac{p_{i1}}{p_{in}})}$$

(3) once $rv_{i1}$ is established, computes the relative values

$rv_{i2}, rv_{i3}, \ldots rv_{ij}, \ldots rv_{in}$ by:

$$rv_{ij} = \frac{p_{1j} rv_{1j}}{p_{ij}}$$

(4) repeats steps (1) through (3) for each evaluator;

(5) derives the total relative value for each alternative j by summing over all evaluators:

$$TRV_j = \sum_{i=1}^{m} rv_{ij}$$

Alternatively, a relative weight, $w_i$ for each evaluator i, determined from the difference between the highest and lowest relative values (as in Fishburn Relative Value) can be computed:

$w_i = (rv_i^* - rv_{i*})$, where $rv_i^*$ is the relative value of the best consequence, evaluator i; and $rv_{i*}$ is the relative value of the worst consequence, evaluator i.

Then, the total relative value for each design j, is obtained from:

$$TRV_j = \sum_{i=1}^{m} w_i rv_{ij}$$

(6) ranks all designs by total relative value.

(7) The program accepts from the user, the names of 2 to 5 designs to be displayed. The computational results: relative value for each alternative with respect to each evaluator, total relative value for each alternative, and the rank of each alternative, are displayed:

| | | |
|---|---|---|
| Designs: | name 1 | name 2 |
| Relative Values: | | |
| goal 1 | $rv_{11}$ | $rv_{12}$ |
| goal 2 | $rv_{21}$ | $rv_{22}$ |
| ...... | | |
| goal m | $rv_{m1}$ | $rv_{m2}$ |
| Total: | $TRV_1$ | $TRV_2$ |
| Ranking: | $r_1$ | $r_2$ |

Both the Case and Fishburn relative value measures are dependent on the set of outcomes defined by the present set of alternatives.

(c) Linear Scoring Function

A linear scoring function (57) requires that all impacts or consequences be assigned a numerical value (possibly

(57) M. L. Manheim et. al.; op. cit. (1969), p. 15.

through transformation from a preference function). Rankings result from computing the weighted sum of evaluator transformed values for each alternative. If the value function is linear with respect to the predicted consequences, the weights can be adjusted, and the total score may be computed directly from multiplying the numerical impacts by their respective weights, and summing over all evaluators. All impacts must be cast in a mode of increasing preference for this latter, simpler form to be used.

The DISCOURSE program "Linear-Scoring" (Figure No. 4.8), assumes the transformation of impact values by a preference function. If the value functions were independently assessed, each evaluator weight $w_i (\sum_i w_i = 1.0)$ represents the tradeoff or substitution rate between evaluators. If the value or utility functions are independent, but each is conditional on the minimum values of each other value function, then each evaluator weight $w_i = 1.0$ (i.e. the value functions are properly scaled so as to incorporate the tradeoff ratios within them). This latter form corresponds to Fishburn's additive utility concept (*). The proper combination of weights with

(*) see discussion on page 42

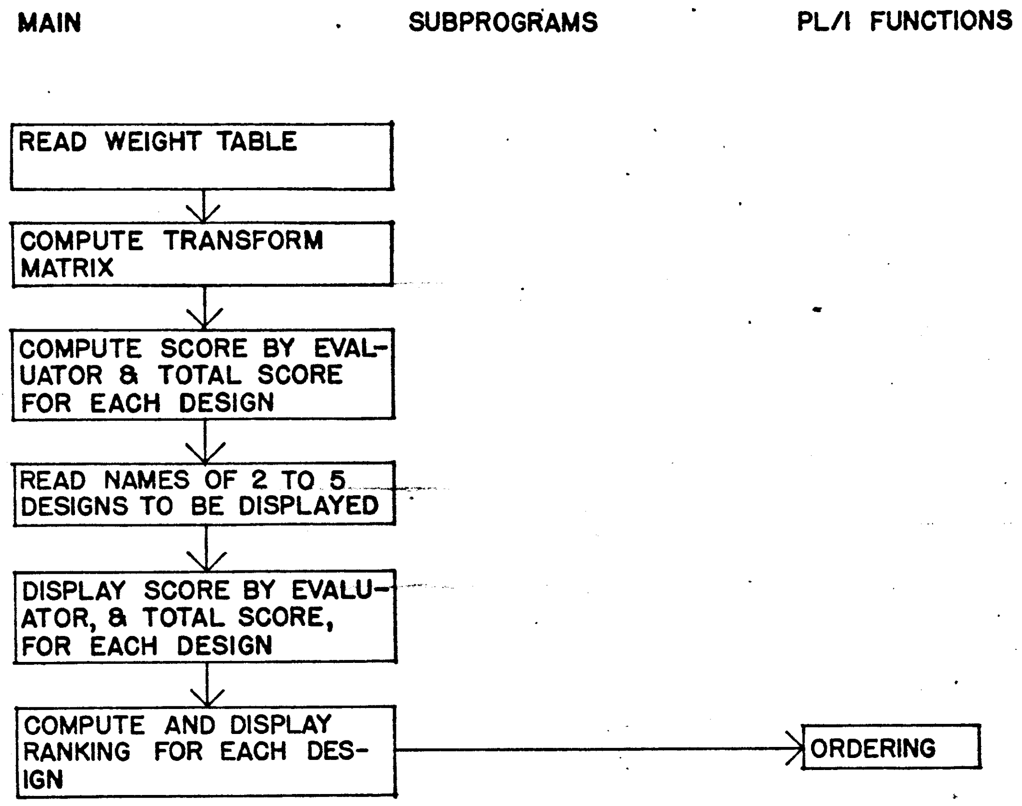

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
READ WEIGHT TABLE
COMPUTE TRANSFORM MATRIX
COMPUTE SCORE BY EVAL-UATOR & TOTAL SCORE FOR EACH DESIGN
READ NAMES OF 2 TO 5 DESIGNS TO BE DISPLAYED
DISPLAY SCORE BY EVALU-ATOR, & TOTAL SCORE, FOR EACH DESIGN
COMPUTE AND DISPLAY RANKING FOR EACH DES-IGN
ORDERING

FIGURE NO. 4·8

value functions must be determined before the program is executed.

The steps in the process are:

(1) The impact matrix of design consequences is mapped onto the value array or function for each evaluator, to yield a transformed matrix:

<u>Impact Matrix</u> <u>Value Array</u> <u>Transform Matrix</u>

$$\begin{array}{c}\\ E_1\\ \vdots\\ E_i\\ \vdots\\ E_m\end{array}\begin{array}{c} A_1 \cdots A_j \cdots A_n\\ \left[\begin{array}{ccc} i_{11} \cdots & i_{1j} & \cdots i_{1n}\\ \vdots & & \vdots\\ & \bigcirc\!\!\!\!\!\!\; i_{ij} & \\ \vdots & & \vdots\\ i_{m1} \cdots & & \cdots i_{mn}\end{array}\right]\\ (i_{ij} = k)\end{array} \rightarrow \begin{array}{c}\\ E_1\\ \vdots\\ E_i\\ \vdots\\ E_m\end{array}\begin{array}{c} 1 \cdots k \cdots r\\ \left[\begin{array}{ccc} v_{11} \cdots & v_{1k} & \cdots v_{1r}\\ \vdots & & \vdots\\ & v_{ik} & \\ \vdots & & \vdots\\ v_{m1} \cdots & & \cdots v_{mr}\end{array}\right]\\ \phantom{x}\end{array} \rightarrow \begin{array}{c}\\ E_1\\ \vdots\\ E_i\\ \vdots\\ E_m\end{array}\begin{array}{c} A_1 \cdots A_j \cdots A_n\\ \left[\begin{array}{ccc} t_{11} \cdots & t_{1j} & \cdots t_{1n}\\ \vdots & & \vdots\\ & t_{ij} & \\ \vdots & & \vdots\\ t_{m1} \cdots & & \cdots t_{mn}\end{array}\right]\\ \phantom{x}\end{array}$$

(2) The score for alternative j with respect to evaluator weight $w_i$, with the transform value $t_{ij}$, i.e.:

$$s_{ij} = w_i t_{ij}$$

<u>Transform Matrix</u> <u>Weights</u> <u>Score Matrix</u>

$$\begin{array}{c}\\ E_1\\ \vdots\\ E_i\\ \vdots\\ E_m\end{array}\begin{array}{c} A_1 \cdots A_j \cdots A_n\\ \left[\begin{array}{ccc} t_{11} \cdots & t_{1j} & \cdots t_{1n}\\ \vdots & & \vdots\\ & t_{ij} & \\ \vdots & & \vdots\\ t_{m1} \cdots & & \cdots t_{mn}\end{array}\right]\end{array} \begin{array}{c}\\ \times\\ \\ \times\\ \\ \times\end{array} \begin{array}{c}\\ \left[\begin{array}{c} w_1\\ \vdots\\ w_i\\ \vdots\\ w_m\end{array}\right]\end{array} \longrightarrow \begin{array}{c}\\ E_1\\ \vdots\\ E_i\\ \vdots\\ E_m\end{array}\begin{array}{c} A_1 \cdots A_j \cdots A_n\\ \left[\begin{array}{ccc} s_{11} \cdots & s_{1j} & \cdots s_{1n}\\ \vdots & & \vdots\\ & s_{ij} & \\ \vdots & & \vdots\\ s_{m1} \cdots & & \cdots s_{mn}\end{array}\right]\end{array}$$

(3) The total score, $TS_i$, for each alternative i, is obtained by summing $s_{ij}$ over all evaluators j; i.e.:

$$TS_i = \sum_{i=1}^{m} s_{ij}$$

(4) Designs are ranked by total score.

(5) The program accepts the name of 2 to 5 designs from the user. Results are displayed in the following format:

| | | | |
|---|---|---|---|
| Designs: | name 1 | name 2 | ... |
| Score: | | | |
| goal 1 | $s_{11}$ | $s_{12}$ | ... |
| goal 2 | $s_{21}$ | $s_{22}$ | ... |
| . | . | . | |
| . | . | . | |
| . | . | . | |
| goal m | $s_{m1}$ | $s_{m2}$ | ... |
| Total: | $TS_1$ | $TS_2$ | ... |
| Ranking: | $r_1$ | $r_2$ | ... |

(d) Quasi-Additive Utility

As discussed above, the quasi-additive form of utility aggregation (58) requires evaluators to be mutually utility independent of each other. This can be tested empirically by determining if the compound preference function of all but one evaluator held fixed, is

(58) R. L. Keeney; op. cit.

dependent only on values of the remaining evaluator. This condition must be satisfied for each evaluator in turn. We showed that the compound utility function for two evaluators, x and y, is:

$$u(x,y) = a_1u_x(x) + a_2u_y(y) + (1-a_1-a_2)u_x(x)u_y(y), \quad (1)$$

where $u_x(x)$ and $u_y(y)$ are utilities in our value array sense, and $a_1$ and $a_2$ are additional "corner" utility assessments which interrelate the two evaluators. For three evaluators, x,y, and z, the form is:

$$\begin{aligned} u(x,y,z) = {} & a_1u_x(x) + a_2u_y(y) + a_3u_z(z) + (b_1-a_1-a_2)u_x(x)u_y(y) \\ & + (b_2-a_1-a_3)u_x(x)u_z(z) + (b_3-a_2-a_3)u_y(y)u_z(z) + \\ & (1-b_1-b_2-b_3+a_1+a_2+a_3)u_x(x)u_y(y)u_z(z). \quad (2) \end{aligned}$$

where $a_1,a_2,a_3,b_2$ and $b_3$ are corner utility assessments.

The DISCOURSE program "Quasi-Separable" (Figure No. 4.9) does not deal with groups of more than three evaluators, since the number of required corner utility assessments goes up rapidly as m, the number of evaluators, increases. (For 4 evaluators, the number required is $2^m-2$, or 14; for 5 evaluators, it is $2^5-2$, or 30; and so on.) For large numbers of grouped evaluators, the additive utility or linear scoring function, is a reasonable approximation to the quasi-additive form.

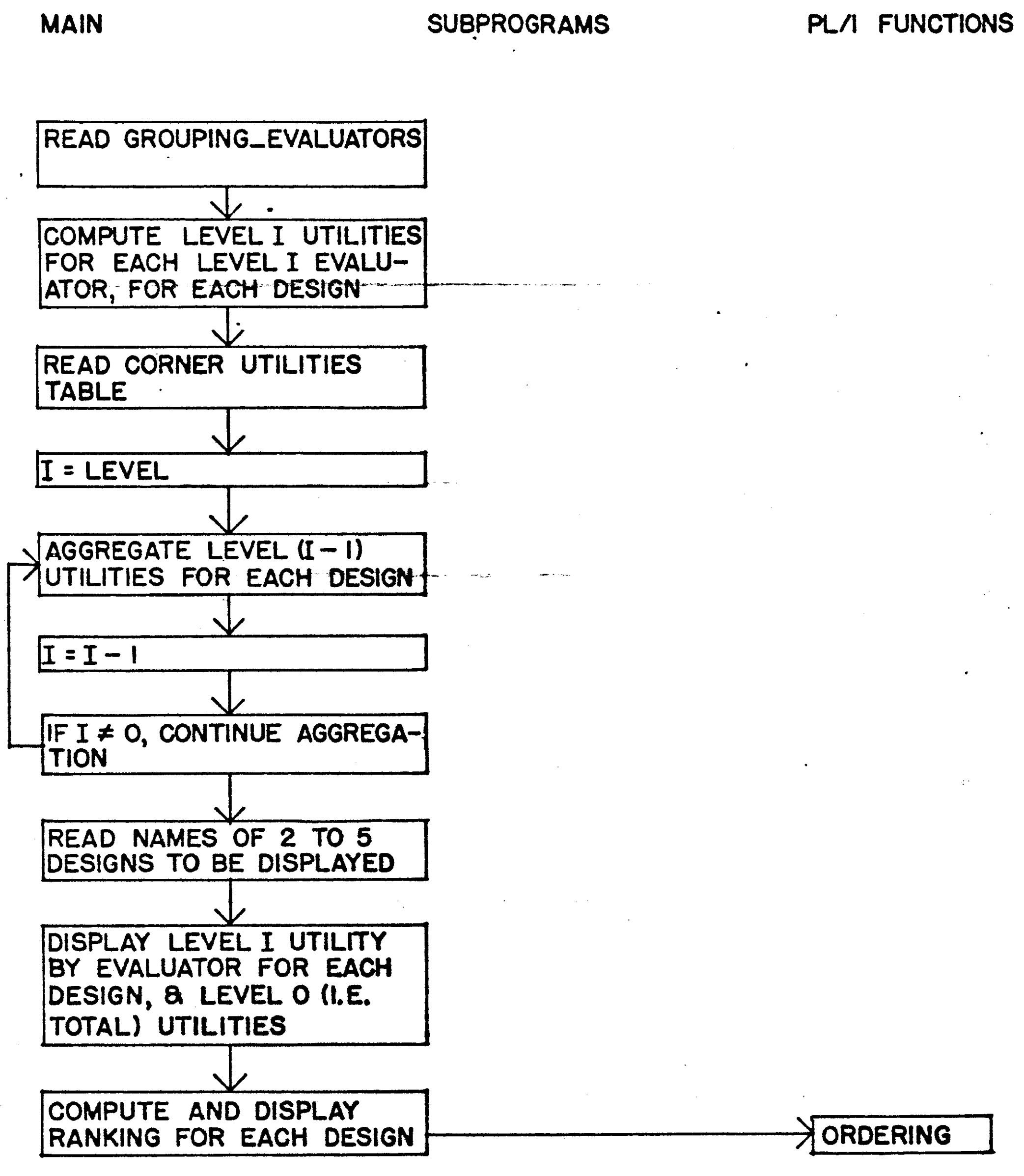

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
READ GROUPING_EVALUATORS
COMPUTE LEVEL I UTILITIES FOR EACH LEVEL I EVALUATOR, FOR EACH DESIGN
READ CORNER UTILITIES TABLE
I = LEVEL
AGGREGATE LEVEL (I – 1) UTILITIES FOR EACH DESIGN
I = I – 1
IF I ≠ 0, CONTINUE AGGREGATION
READ NAMES OF 2 TO 5 DESIGNS TO BE DISPLAYED
DISPLAY LEVEL I UTILITY BY EVALUATOR FOR EACH DESIGN, & LEVEL 0 (I.E. TOTAL) UTILITIES
COMPUTE AND DISPLAY RANKING FOR EACH DESIGN
ORDERING

FIGURE NO. 4·9

For quasi-additive utility assessment, a goal structure must be input, since the program must determine which groups of evaluators are to be aggregated (which will vary from problem to problem), and retrieve the appropriate corner utility values. Qualitatively, the procedure is as follows:

(1) The impact matrix of design consequences for level i is mapped onto the value array or function for each evaluator at level i, to yield a transformed matrix of single-evaluator utilities:

<u>Impact Matrix</u> <u>Value Array</u> <u>Transform Matrix</u>

$$\begin{array}{cc} & \begin{array}{ccccc} A_1 & \cdots & A_j & \cdots & A_n \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_i \\ \vdots \\ E_m \end{array} & \begin{bmatrix} i_{11} & \cdots & i_{1j} & \cdots & i_{1n} \\ \vdots & & & & \vdots \\ & & i_{ij} & & \\ \vdots & & & & \vdots \\ i_{m1} & \cdots & & \cdots & i_{mn} \end{bmatrix} \end{array} \longrightarrow \begin{array}{cc} & \begin{array}{ccccc} 1 & \cdots & k & \cdots & r \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_j \\ \vdots \\ E_m \end{array} & \begin{bmatrix} v_{11} & \cdots & v_{1k} & \cdots & v_{1r} \\ \vdots & & & & \vdots \\ & & v_{ik} & & \\ \vdots & & & & \vdots \\ v_{m1} & \cdots & & \cdots & v_{mr} \end{bmatrix} \end{array} \longrightarrow \begin{array}{cc} & \begin{array}{ccccc} A_1 & \cdots & A_j & \cdots & A_n \end{array} \\ \begin{array}{c} E_1 \\ \vdots \\ E_j \\ \vdots \\ E_m \end{array} & \begin{bmatrix} t_{11} & \cdots & t_{1j} & \cdots & t_{1n} \\ \vdots & & & & \vdots \\ & & t_{ij} & & \\ \vdots & & & & \vdots \\ t_{m1} & \cdots & & \cdots & t_{mn} \end{bmatrix} \end{array}$$

$(i_{ij} = k)$

(2) For each group of evaluators at level i, the program determines from the number in each group, whether formula (1) or (2) (above) applies, or whether the utility can simply be transferred to the next level. The required corner utility values are retrieved as each set of evaluators is examined in turn, and the

compound group utility for level i-1 is computed.

(3) Step (2) is repeated for each level of aggregation, until a single multi-dimensional utility assessment, u(0,1,j), results. (First subscript = level, second = group designation, third = alternative designation). For each intermediate level, the grouping of goal variables in the structure must be determined, and the necessary computations performed, dependent on that grouping.

(4) Steps (2) and (3) are repeated for each design.

(5) Designs are ranked by level 0 utility; u(0,1,j).

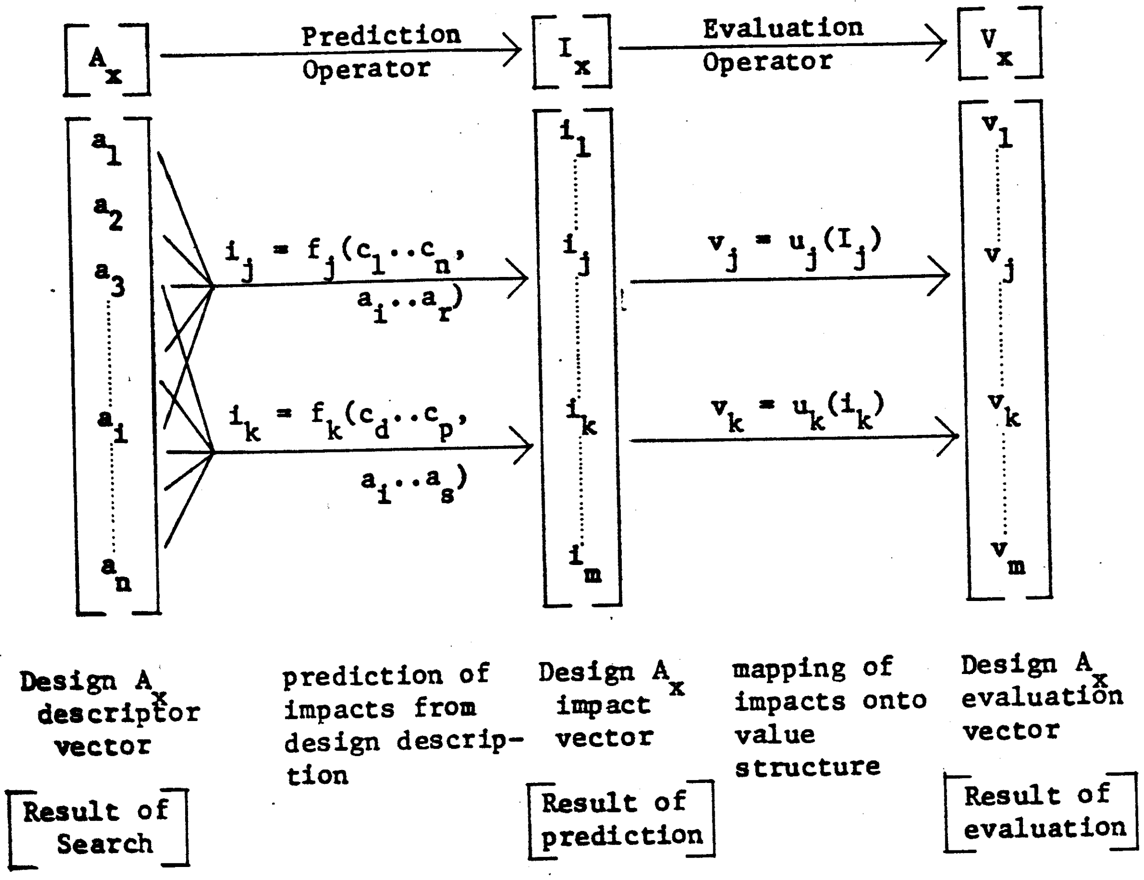
(6) The program accepts the names of 2 to 5 designs for display. The computational results for level i utilities, level 0 aggregated utility, and ranking for each design, are shown:

| Designs: | name 1 | name 2 | ... |
|---|---|---|---|
| Utilities Level i: | | | |
| goal 1 | $t_{11}$ | $t_{12}$ | ... |
| goal 2 | $t_{21}$ | $t_{22}$ | ... |
| . | . | . | |
| . | . | . | |
| . | . | . | |
| goal m | $t_{m1}$ | $t_{m2}$ | ... |
| Aggregated Utility: | u(0,1,1) | u(1,1,2) | ... |
| Ranking | $r_1$ | $r_2$ | |

## 4.3 Dynamic Evaluation Model

The Dynamic Evaluation Model uses the concept of hierarchically structured levels for both goals ("evaluators") and actions ("design alternatives"). In the hierarchy of evaluators, upper level general goals are explicated, specified, or clarified by lower level objectives. Lower level goals are components therefore, or parent goal vectors. In the hierarchy of description of design alternatives, lower level designs may be seen as variations within the partial constraints of their parent, least including designs. These more detailed alternatives do not explicate or specify what is meant by the descriptors of parent designs; they supply attributes, or attribute values left undefined by the metric of the immediately preceding level.

The relationships between the evaluator language and the descriptor language are complex. Computationally, we would prefer each of the structures to be internally simple; however, in so constructing them, we make the external relations between the hierarchies very complex, and analysable only probabilistically, if at all. Figure No. 4.10 illustrates this relationship between description and evaluation:


$[A_x]$
Prediction Operator
$[I_x]$
Evaluation Operator
$[V_x]$
$a_1, a_2, a_3, \ldots, a_i, \ldots, a_n$
$i_j = f_j(c_1 \ldots c_n, a_i \ldots a_r)$
$i_k = f_k(c_d \ldots c_p, a_i \ldots a_s)$
$i_1, \ldots, i_j, \ldots, i_k, \ldots, i_m$
$v_j = u_j(I_j)$
$v_k = u_k(i_k)$
$v_1, \ldots, v_j, \ldots, v_k, \ldots, v_m$
Design $A_x$ descriptor vector
Result of Search
prediction of impacts from design description
Design $A_x$ impact vector
Result of prediction
mapping of impacts onto value structure
Design $A_x$ evaluation vector
Result of evaluation

$C = (c_d, \ldots c_p)$ is the set of context variables.

Suppose for example, for the design alternative $A_x$:

descriptor $a_i$ is a variable "type of construction";

function $f_j(c_1, \ldots c_n, a_i \ldots a_r)$ is the prediction of unit rentals;

function $f_k(c_d, \ldots c_p, a_i, \ldots a_s)$ is the prediction of building maintenance costs;

and the resulting valuations are:

$v_j$ = low (i.e. high rentals);

$v_k$ = high (i.e. low maintenance costs).

If we attempt to improve $A_x$ by trying to find some way of increasing $v_j$(i.e. lowering rents) while increasing or maintaining the same $v_k$ (i.e. the same or lower maintenance costs), one of the possible design variables we could vary would be $a_i$ (for example, change type of construction from concrete to wood). The resulting designs thus generated might result in an increase in $v_j$ (i.e. lower rentals), but simultaneously, a decrease in $v_k$ (i.e. higher building maintenance costs). By varying the components of a set of variables in one domain, we cannot directly and with certainty, infer the impact effect on the other domain without going through the complex prediction, and then evaluation mappings. However, within each domain (goal or action) we may have relative freedom to manipulate subsets of the overall variable set independently of other subsets.

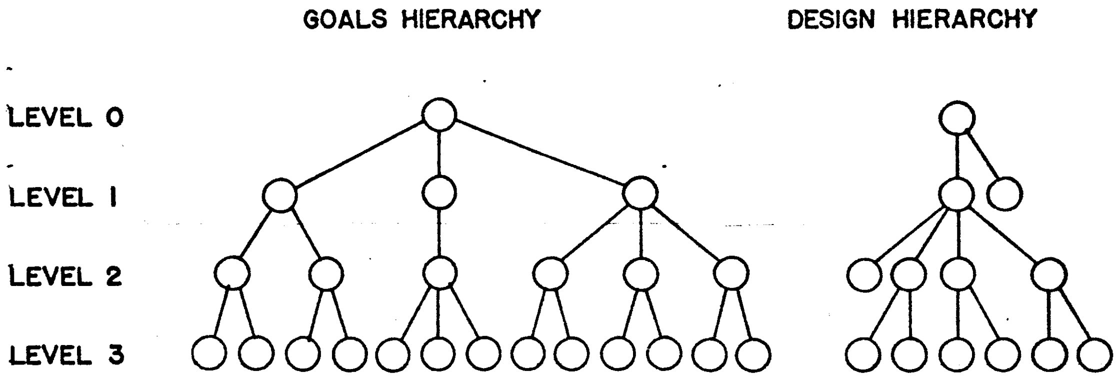
The model assumes that relatively simple goal and design hierarchies can be constructed, and will remain relatively stable over the planning process. In particular, we require a goals hierarchy which can be formulated at least in conditional utility independent form, and a design hierarchy with well-defined transitions from level to level, and inclusion relationships from "parent" to "offspring" designs.

Furthermore, at least one goal level can be found to correspond to each level in the design structure. The converse is not necessarily true: there may be more goal levels than design levels if some of the intermediate design metrics are not sufficiently so as to make another level of search (and therefore, prediction and evaluation) worthwhile in terms of additional information gained. On the other hand, intermediate aggregations of goal variables may be useful in the extension of the goal tree. For simplicity, we assume goal and action levels coincide. In well-defined problems, where searches and tests are embedded within each algorithm, there is a direct correspondence between action and goal levels. There cannot be more action levels than goal levels, since this would imply the generation of actions which cannot be compared or evaluated on the same level, which goes against the notion of level as a metric- or measurement-based concept. (cf. Figure No. 4.11).

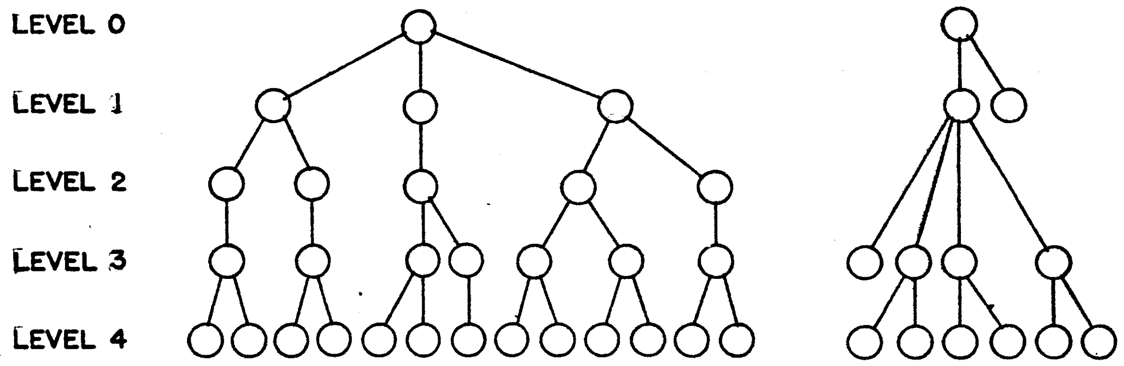
The purpose of the evaluation model is to guide the designer through the planning process, in suggesting which experiments to undertake, and using the accumulated history (in terms of experiential knowledge) of the process as a guide to future action. The model also allows the designer to draw some inferences about the nature of experiments, given by the description field of the system. An example will illustrate this point:

After the initialization of a planning process, the generation of the first alternative and its descriptor set, is arbitrary. (The "universal design" is not really an alternative, but rather a vehicle for the initial subjective distribution over expected evaluations emanating from it).

1. SIMPLE COINCIDENCE



2. MORE GOAL LEVELS THAN DESIGN LEVELS



3. NOT ALLOWED: MORE DESIGN LEVELS THAN GOAL LEVELS

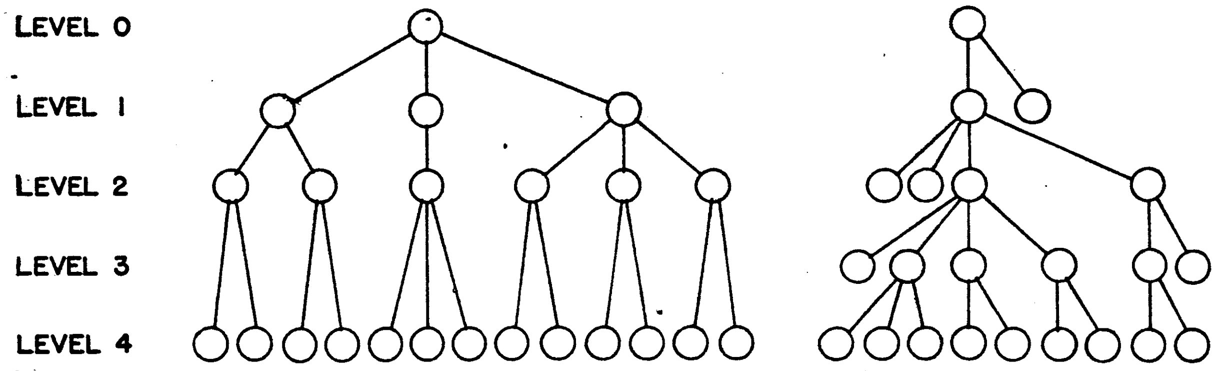


FIGURE NO. 4·11

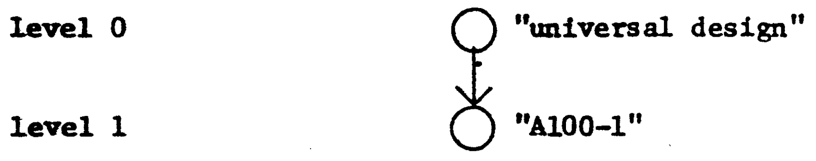


Figure No. 4.12

However, after the generation of this first arbitrary design, subsequent designs are constrained and influenced by the results of the process. For a second pass, we would have:

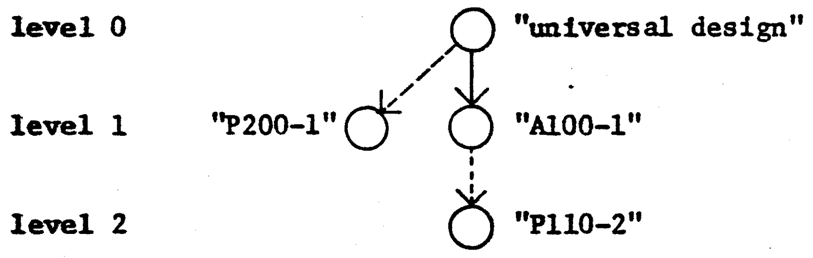


Figure No. 4.13

where "P200-1" and "P110-2" are potential design experiments, not yet executed. Potential design P200-1, on the same level as executed design A100-1, is constrained in several senses:

(a) it must possess the same descriptor attributes as A100-1, at least in terms of those relevant to evaluation, so that A100-1 and A200-1 (if executed) may be meaningfully compared;

(b) on the other hand, it must search out a different portion of the solution space from A100-1, in terms of descriptors appropriate to that level. That is, it cannot be so similar to A100-1 that some lower-level design, say Axx0-2, could be developed from both A100-1, and A200-1 as parents. Different parent designs imply distinctly different "offspring" designs if the concept of metric is to have any meaning.

Learning from the results of the planning process covers a number of other areas as well; for example, the perceived characteristics of levels, and single-level operations, change. Initially we require that a design at level j be developed from an existing parent at level j-1. In other words, a jump to detailed building configurations without having executed the parent land use plan, is not allowed. Figure No. 4.14 illustrates this point:

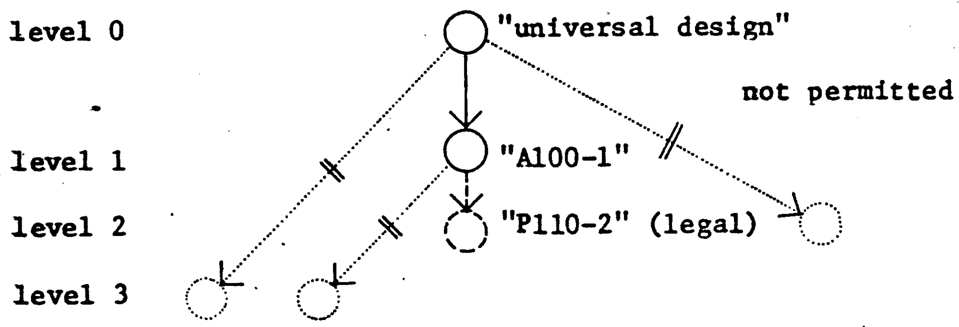


Figure No. 4.14

The reasoning behind this restriction, is that the designer cannot skip levels of description until he has learned about the kinds of information that may be generated at each level, particularly when attributes of upper-level alternatives serve as partial constraints for lower levels.

As the planning process continues, learning about the nature of the solution space at each level improves; certain attribute sets are perceived as being crucial to solutions at that level; other avenues of exploration are cut off as alternatives at that level experiment with portions of the solution space. Therefore, we would expect the cost of generating al alternative at a particular level to decrease over the history of the process, since the unexplored space becomes progressively smaller (and the costs of search are related to the area and density of that space). This is especially true for alternatives generated from a parent which has already produced "brother" designs at that level; for example:

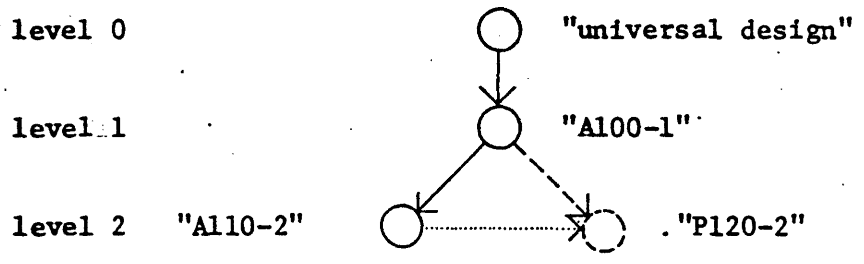


Figure No. 4.15

It should cost less to generate A120-2 than it was to generate A110-2 because in many respects A120-2 will be similar to its "brother"; in fact, it may be an incremental variant of A110-2, suggested by the latter's evaluation, which identified the strong and weak points of that design. Invocation of user operations such as "satisfaction", or its proposed extensions, may point the way to incremental improvement of executed designs, by identifying goal variables which are poorly satisfied (where design effort should be concentrated) and goal variables which are well satisfied (which may either be loosened, or point to design variables which should be held fixed from experiment to experiment on that level).

Later on in the process, multi-level jumps of the kind restricted earlier could be allowed; for example:

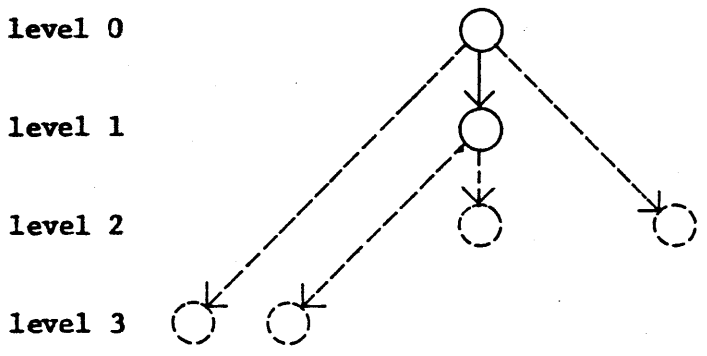


Figure No. 4.16

because the designer has learned enough about the kind of information acquired from intervening levels, so as to be able to dispense with them, and economize on his search effort. However, in its present implementation, the model does not allow such jumps, (and therefore loses whatever heuristic value there may be in first exploring lower levels as a means to improving search performance at intermediate levels).

(a) Components:

(i) Goal Structure:

(1) A set of evaluators decomposed in a goal fabric, with level and inclusion relationships, must be defined(*). The computer program "Evaluators" accepts the names and level designation of evaluators from the user at the console, and generates an array "structure-goals" which defines these level and inclusion relationships for use by subsequent programs.
(2) a value array table (step utility function) or preference function for each elemental goal variable.
(3) a table of corner utilities or tradeoffs between subsets of goal variables for each level.
Figure No. 4.17 represents the goal structure for the M.I.T. North West Area Project, an illustrative application; Figure No. 4.18 illustrates how this structure is stored in the Discourse array "structure-

(*) a description of what is meant by goal decomposition is given, on page

GOAL STRUCTURE – M.I.T. NORTH WEST AREA PROJECT

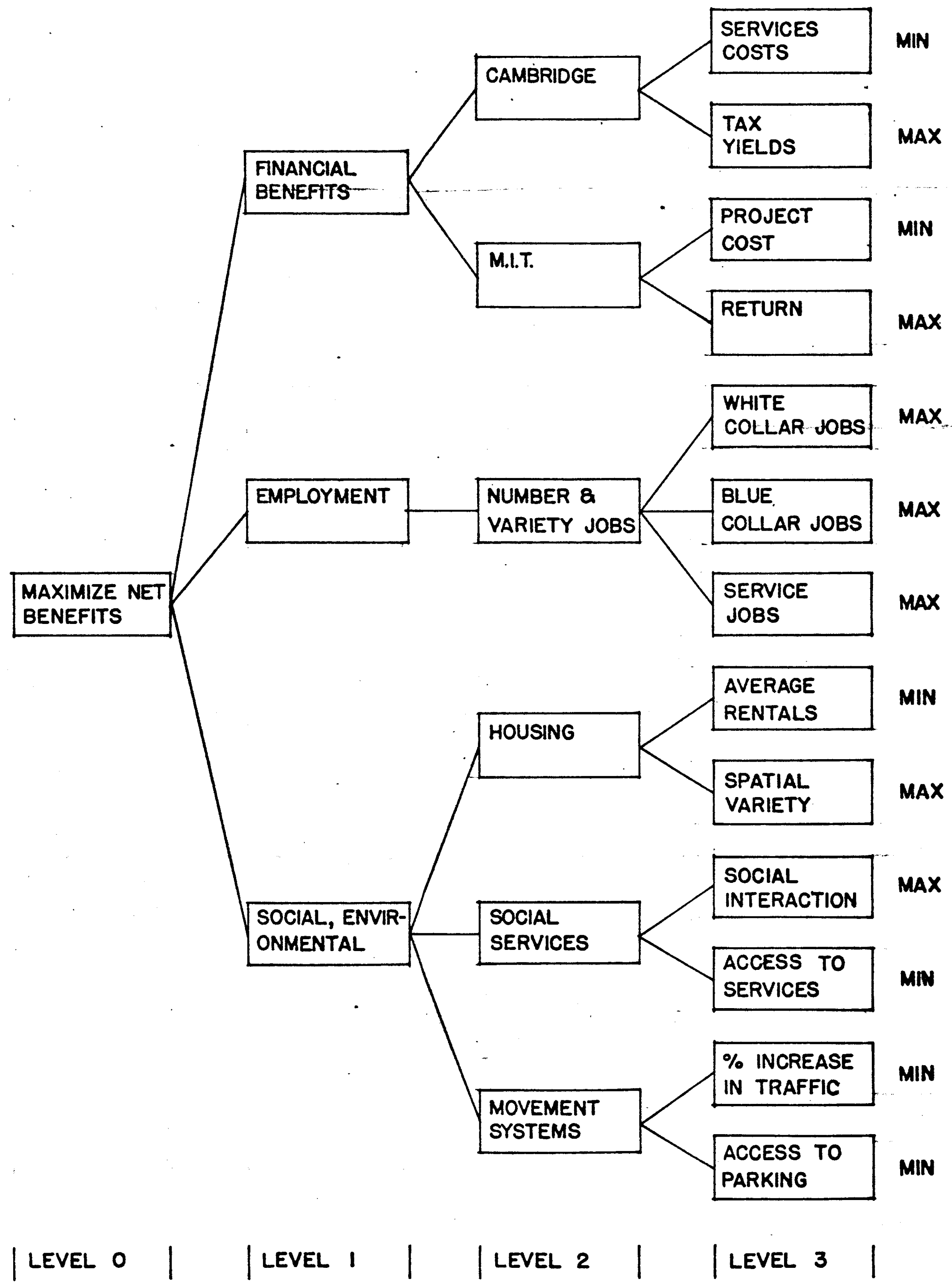
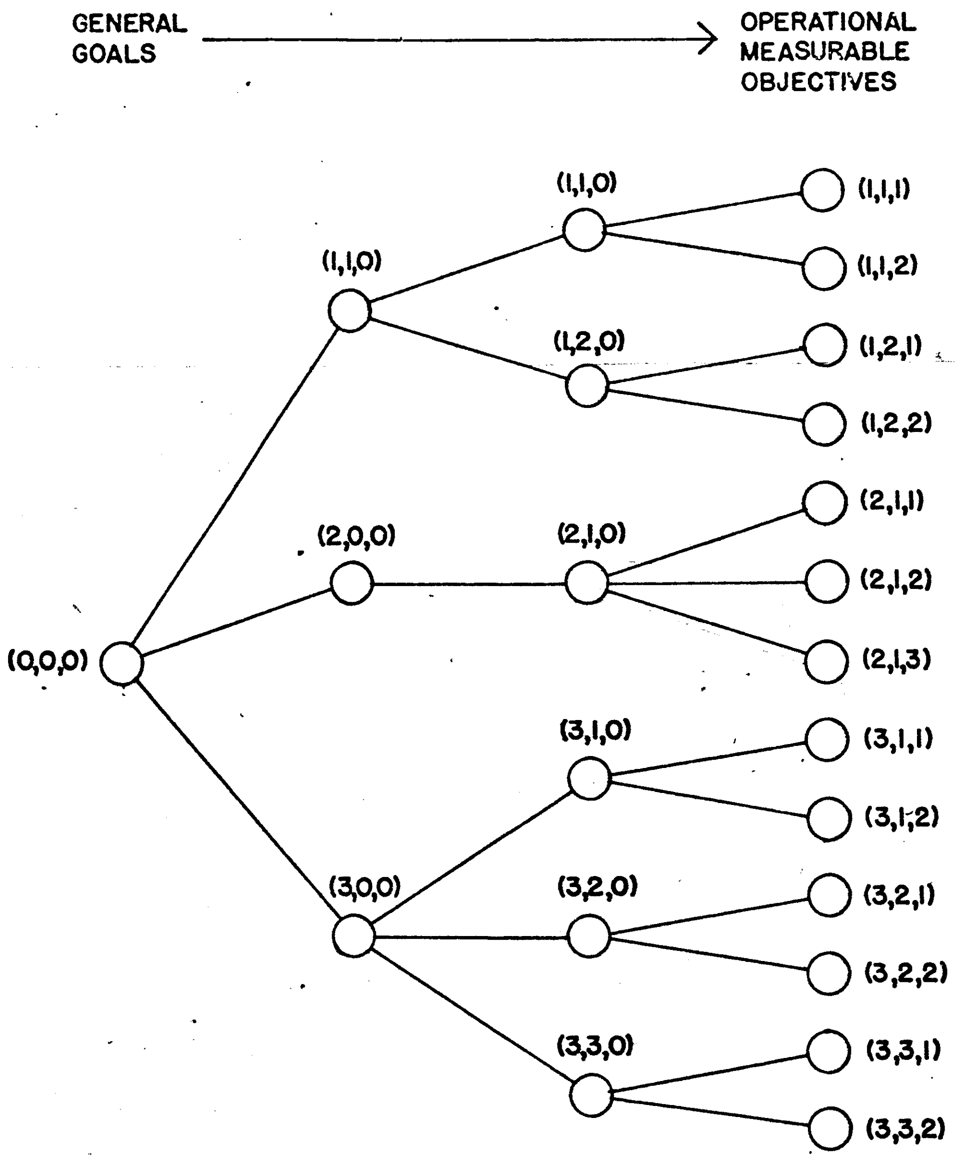


FIGURE NO. 4.17

## REPRESENTATION OF GOAL STRUCTURE



| | | | |
|---|---|---|---|
| LEVEL 0 | LEVEL 1 | LEVEL 2 | LEVEL 3 |
| U(0,1,K) | U(1,J,K) | U(2,J,K) | U(3,J,K) |
| J = 1 | J = 1 TO 3 | J = 1 TO 6 | J = 1 TO 13 |
| | GROUP(1,1) = 3 | GROUP(2,1) = 2 | GROUP(3,1) = 2 |
| | | GROUP(2,2) = 1 | GROUP(3,2) = 2 |
| | | GROUP(2,3) = 3 | GROUP(3,3) = 3 |
| | | | GROUP(3,4) = 2 |
| | | | GROUP(3,5) = 2 |
| | | | GROUP(3,6) = 2 |

J = GROUP DESIGNATION
K = DESIGN DESIGNATION (ORDER OF EXECUTION)
GROUP(LEVEL,J) = NUMBER OF GOALS IN A GROUP

FIGURE NO. 4·18

goals", as well as in other arrays.

(ii) Design Structure:

(1) A set of levels to which each level of the Goal Structure corresponds, is defined. In the DISCOURSE system, which comprises a data structure for spatial metrics, the definition of level is possibly simpler than for non spatially-oriented computer systems, since the consistent progression of scaled representations (usually by factors of 2) is commonly accepted practice in architecture and planning.

For the M.I.T. North West Area Project, an arbitrary, though useful scale factor of 4 defines levels and their associated metrics: each grid cell in a lower level metric is 1/4 the size of a grid cell in the immediately preceding level: (cf. Figure No. 4.19).

level 1: scale: 200' x 240'

level 2: scale: 100' x 120'

level 3: scale: 50' x 60'

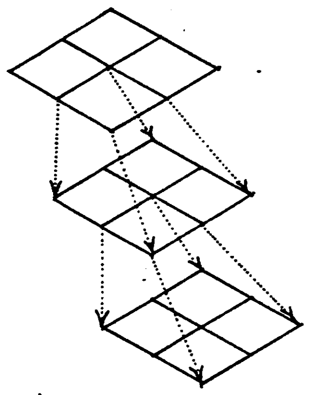


Figure No. 4.19

The advantages of this definition are:

(1) through progressive scale factors of 4, any desired level of detail can be reached in a small number of steps;

(2) the scale factor relates to commonly accepted practice;

(3) the influence of attributes as partial constraints from higher levels to lower levels; as well as the aggregation of values from lower level grid cells to higher level grid cells, can be easily done through a pointer system which references the different metrics to each other. Figure No. 4.20 shows this metric inclusion:

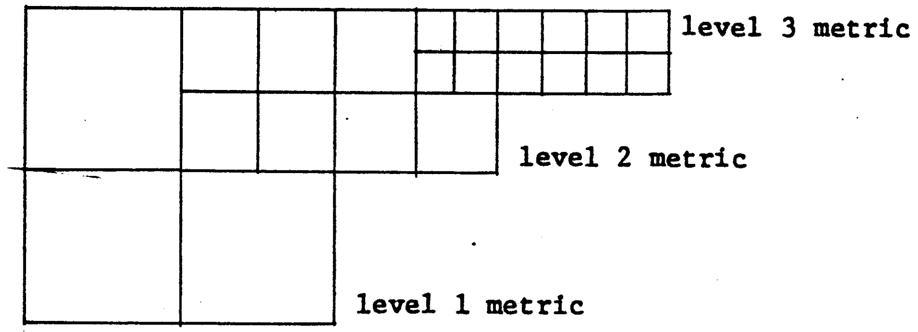


Figure No. 4.20

This referencing is not easily done if the metrics are not so aligned and consistently scaled, for example (cf. Figure No. 4.21):

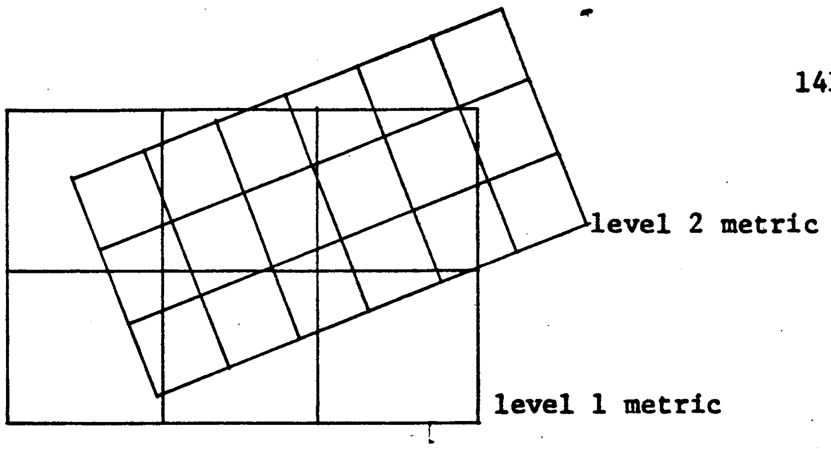


Figure No. 4.21

To accept this latter representation implies that the spatial disposition of attributes is not as important as some other distinction or scaling in defining metrics.

(2) a set of level and inclusion relationships for each new action (generated internally by the Discourse program "Dynamic-Control", and stored in the array "structure-designs".

(3) a list of names for each new design (accepted from the user).

(4) an impact vector by appropriate level evaluators, for each new design.

Figure No. 4.22 illustrates the Discourse array representation of a hypothetical terminal design structure;

(iii) Probability Distributions:

(1) current value distribution:

REPRESENTATION OF TERMINAL DESIGN STRUCTURE

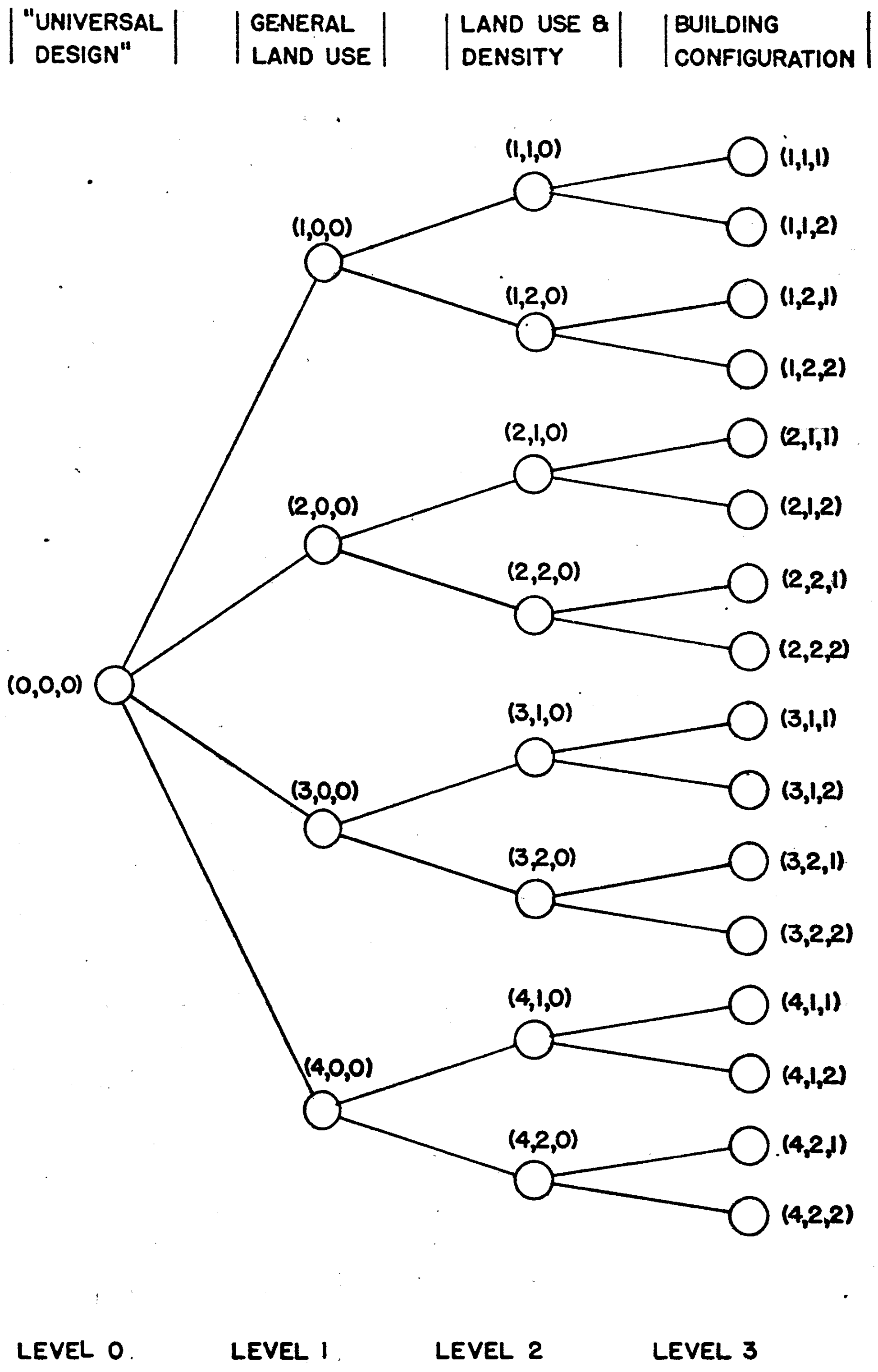


FIGURE NO. 4·22

$f''_k(\theta)$ for each executed design k;

(2) Single Level Operator distributions:

$h_i(u(0,1,k) - \theta)$, for each level i.

(b) <u>Steps in the Hierarchical Planning Model</u>

(1) Determine the legal potential design experiments from the current design structure. Given the restriction on the derivation of experiments, the number of potential experiments will never exceed the current number of executed designs. An experiment is defined as the application of a single level (i+1) search-selection operator to a current design, level i, to yield a new design level i+1.

(2) Compute the expected prior utilities for all potential experiments, and select the experiment k, with the highest utility for implementation:

$E(u(0,1,k)) = \sum_y p_{k_p}(y) \cdot u(k,y)$ where $k_p$ is the immediate parent of experiment k.

$$u(k,y) = u(0,1,k^*) \text{ where } k^* \text{ is the best current elemental action if } y < u(0,1,k^*)$$

$$= u(0,1,k_p) \text{ if } y > u(0,1,k^*)$$

(3) Test whether the expected improvement from the best design experiment over the current best elemental design (if it exists) is greater than the threshold criterion. If it is not, stop the process.

(4) Generate the new chosen design k, predict its appropriate level i impacts, and store the impact vector $I_k = (i_{ik},\ldots i_{mk})$ in the current impact file. (This step may transfer out of the computer environment, or to another set of computer-based, project-dependent routines)

(5) Compute the level i utility u(i,j,k), for each appropriate evaluator j, for the executed design, k.

(6) Tracing through the goal tree level by level, aggregate utilities to compute the level 0 utility, u(0,1,k) for the new design.

(7) Revise the prior current values over the generated design, and its including designs, by Bayes' Theorem:

for design k and its parent:

$$f''_k(\theta/u(0,1,k)) = \frac{f'_k(\theta).h_i(u(0,1,k)-\theta)}{\sum_{\theta} f'_k(\theta).h_i(u(0,1,k)-\theta)}$$

and similarly for the remaining including designs. This revision is performed even if new design k is at the lowest level, since we assume a probabilistic interpretation of utilities at all levels, arising from possible errors in defining evaluators.

(8) If the generated design k was an elemental design, compute its expected value, and revise the curmet rankings of elemental designs.

(9) Return to Step (1) and repeat the process for the next cycle.

This logic is followed in the DISCOURSE program "Dynamic-Control".

(c) Implementation Restrictions

(1) Potential design experiments: a design experiment is legal only if its immediate parent, least including design has been generated. The reasoning behind this has been discussed above. The number of potential designs will always be less than, or equal to the current number of executed designs. Figure No. 4.23 illustrates the effect of this restriction in a hypothetical design process development.

(2) Single Stage analysis: only single design experiments are examined, not strategies of two or more experiments. This restriction is primarily to reduce computation.

(3) Operator characteristics: the $h_i(u(0,1,k)-\theta)$ function is assumed constant over all $\theta$, and over the history

HYPOTHETICAL DESIGN PROCESS DEVELOPMENT

◯ = DESIGNS ALREADY GENERATED
◌ = POTENTIAL DESIGN EXPERIMENTS

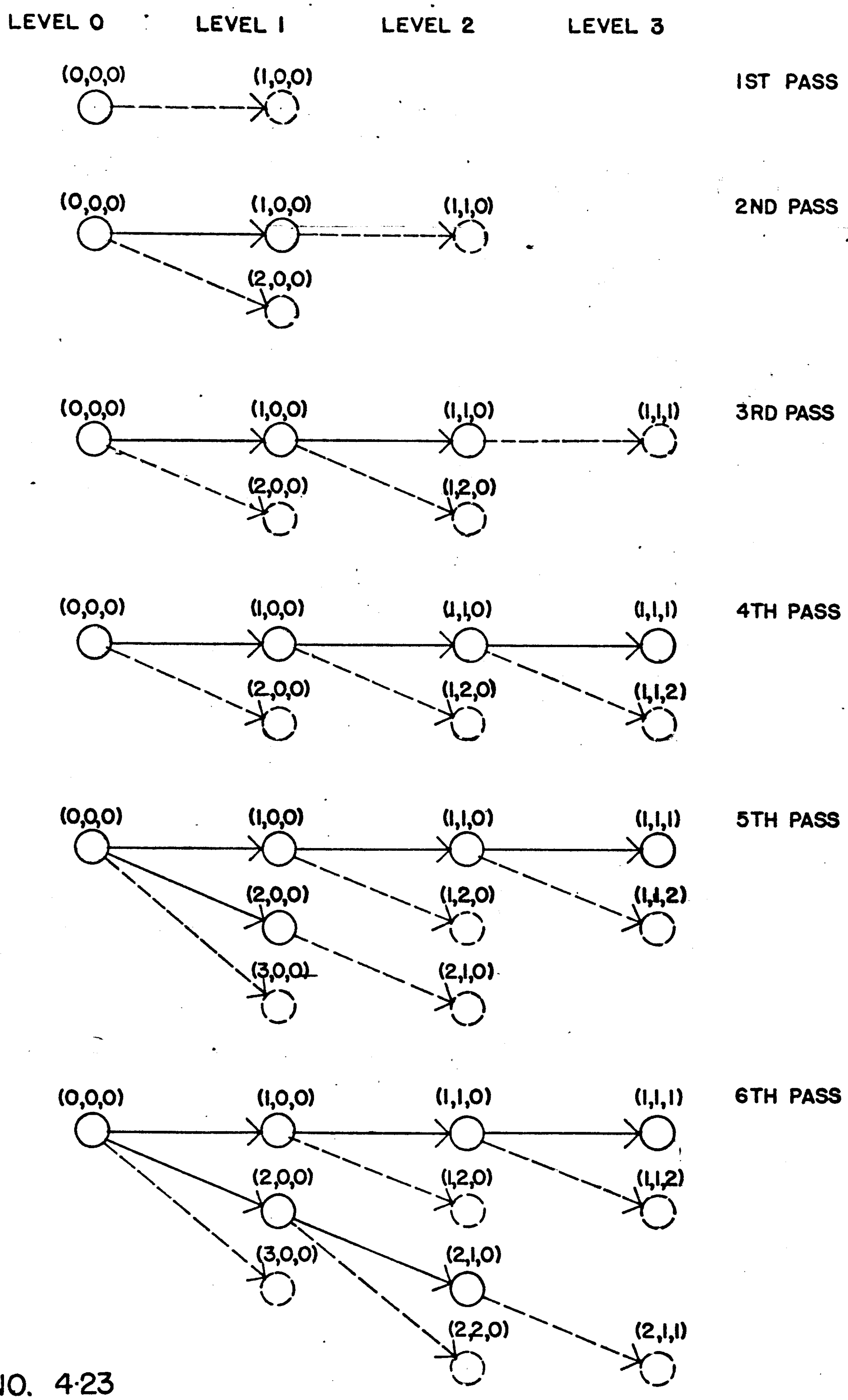


FIGURE NO. 4·23

of the process; for each level i. No revision of operator characteristics is implemented.

(4) Cost of experiments: are not included in the utility calculations, since they are assumed to be highly variable (as discussed above). A threshold criterion for improvement, is substituted.

(d) Implementation

The Dynamic Evaluation model comprises a set of programs, one of which, "Dynamic-Control", is executed after every new design alternative has been generated and its predicted consequences stored in the current impact file. Overall Program Control is illustrated in Figure No. 4.24. A process flow chart, with associated project information file retrieval and updating, is shown in Figure No. 4.25.

Execution proceeds as follows:

(1) Initially, the DISCOURSE program "Evaluators" is executed to accept evaluator names from the console, and derive level and inclusion relationships among evaluators.

Formal program control begins after the first mandatory design experiment, A100-1, has been generated, and assessed.

(2) The DISCOURSE program, "Dynamic-Control-1" (Figure No. 4.26) is called only once since it has only a subset of the functions of the main program, Dynamic-

Control. It determines potential design experiments, computes their expected values from the priors of their immediate parents, and selects the best experiment for implementation, through calling the sub-program, "selection" (Figure No. 4.27). The program accepts a name for the new design from the user, transfers its status from "potential" to "current" and outputs updated parameters.

(3) Subsequently, after any design experiment has been generated, and its predicted impacts stored in the current impact file, the DISCOURSE program "Dynamic-Control", (Figure No. 4.28), is executed. Since there probabily will have been a transfer out of the computer environment preceding this, the program first reads in a number of information files, such as the preference structure, current impact matrix, design and goal structures, and program parameters. Following this, it then:

(a) computes the utility of the new design with respect to each evaluator at the appropriate level;

(b) aggregates utilities in the quasi-additive form to derive an overall utility for the alternative;

(c) revises prior distributions over the new design and its including designs, through calling subprogram "Bayes-Posterior" (Figure No. 4.29);

PROGRAM FLOW CHART

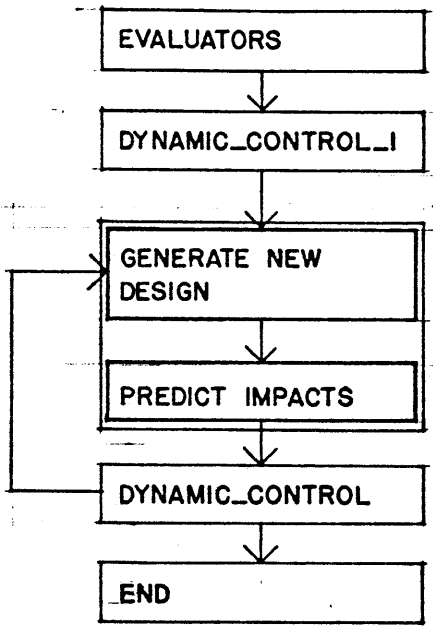


FIGURE NO. 4·24

PROCESS FLOW CHART

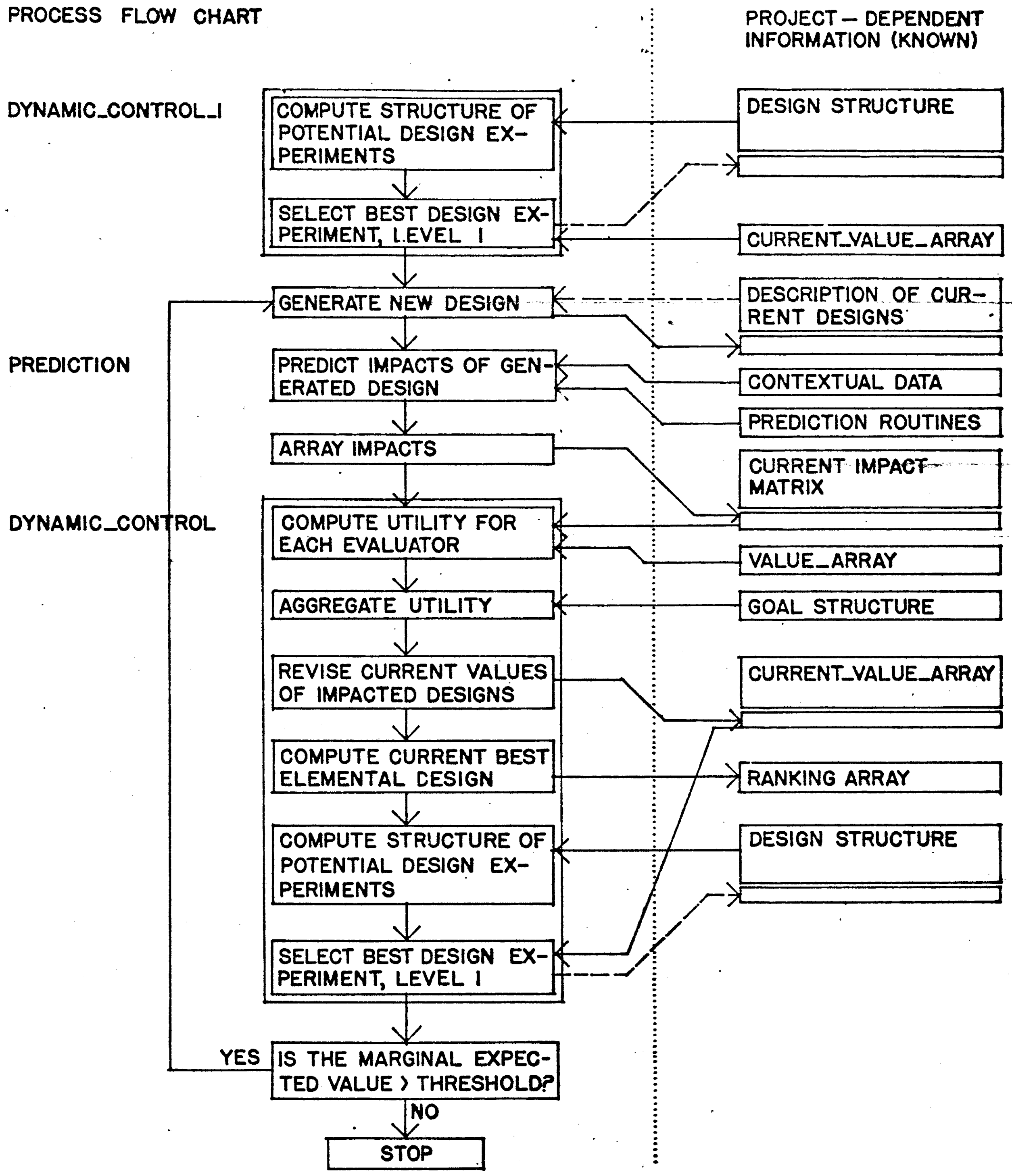

PROJECT — DEPENDENT INFORMATION (KNOWN)
DYNAMIC_CONTROL_I
COMPUTE STRUCTURE OF POTENTIAL DESIGN EXPERIMENTS
SELECT BEST DESIGN EXPERIMENT, LEVEL I
DESIGN STRUCTURE
CURRENT_VALUE_ARRAY
GENERATE NEW DESIGN
DESCRIPTION OF CURRENT DESIGNS
PREDICTION
PREDICT IMPACTS OF GENERATED DESIGN
CONTEXTUAL DATA
PREDICTION ROUTINES
ARRAY IMPACTS
CURRENT IMPACT MATRIX
DYNAMIC_CONTROL
COMPUTE UTILITY FOR EACH EVALUATOR
VALUE_ARRAY
AGGREGATE UTILITY
GOAL STRUCTURE
REVISE CURRENT VALUES OF IMPACTED DESIGNS
CURRENT_VALUE_ARRAY
COMPUTE CURRENT BEST ELEMENTAL DESIGN
RANKING ARRAY
COMPUTE STRUCTURE OF POTENTIAL DESIGN EXPERIMENTS
DESIGN STRUCTURE
SELECT BEST DESIGN EXPERIMENT, LEVEL I
YES
IS THE MARGINAL EXPECTED VALUE > THRESHOLD?
NO
STOP

FIGURE NO. 4·25

DYNAMIC_CONTROL_I (DISCOURSE)

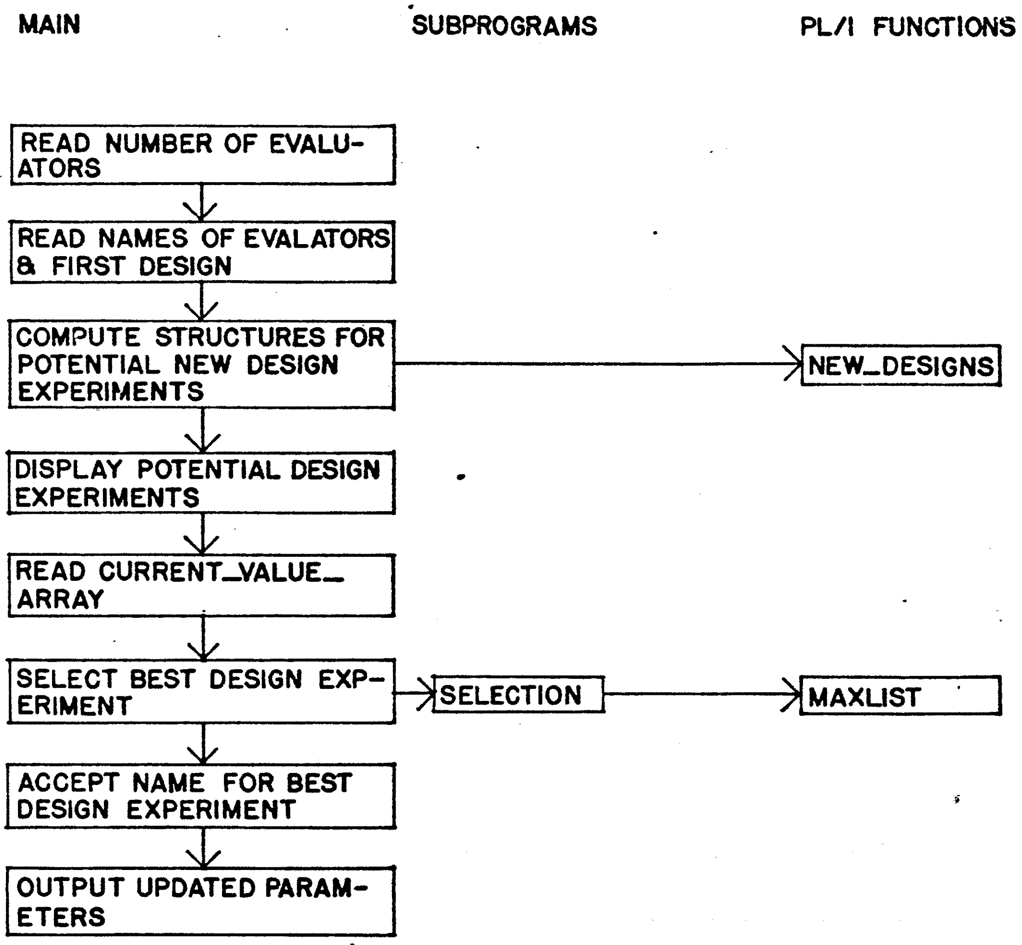

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
READ NUMBER OF EVALUATORS
READ NAMES OF EVALATORS & FIRST DESIGN
COMPUTE STRUCTURES FOR POTENTIAL NEW DESIGN EXPERIMENTS
NEW_DESIGNS
DISPLAY POTENTIAL DESIGN EXPERIMENTS
READ CURRENT_VALUE_ARRAY
SELECT BEST DESIGN EXPERIMENT
SELECTION
MAXLIST
ACCEPT NAME FOR BEST DESIGN EXPERIMENT
OUTPUT UPDATED PARAMETERS

FIGURE NO. 4·26

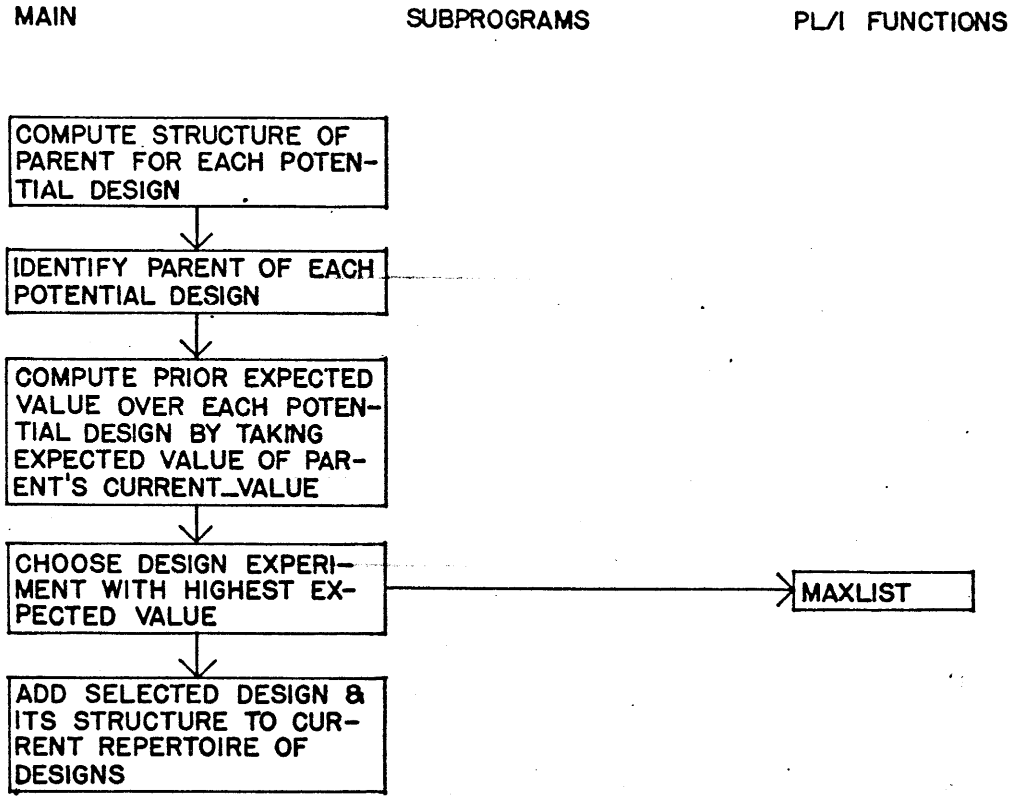

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
COMPUTE STRUCTURE OF PARENT FOR EACH POTENTIAL DESIGN
IDENTIFY PARENT OF EACH POTENTIAL DESIGN
COMPUTE PRIOR EXPECTED VALUE OVER EACH POTENTIAL DESIGN BY TAKING EXPECTED VALUE OF PARENT'S CURRENT_VALUE
CHOOSE DESIGN EXPERIMENT WITH HIGHEST EXPECTED VALUE
MAXLIST
ADD SELECTED DESIGN & ITS STRUCTURE TO CURRENT REPERTOIRE OF DESIGNS

FIGURE NO. 4·27

DYNAMIC_CONTROL (DISCOURSE)

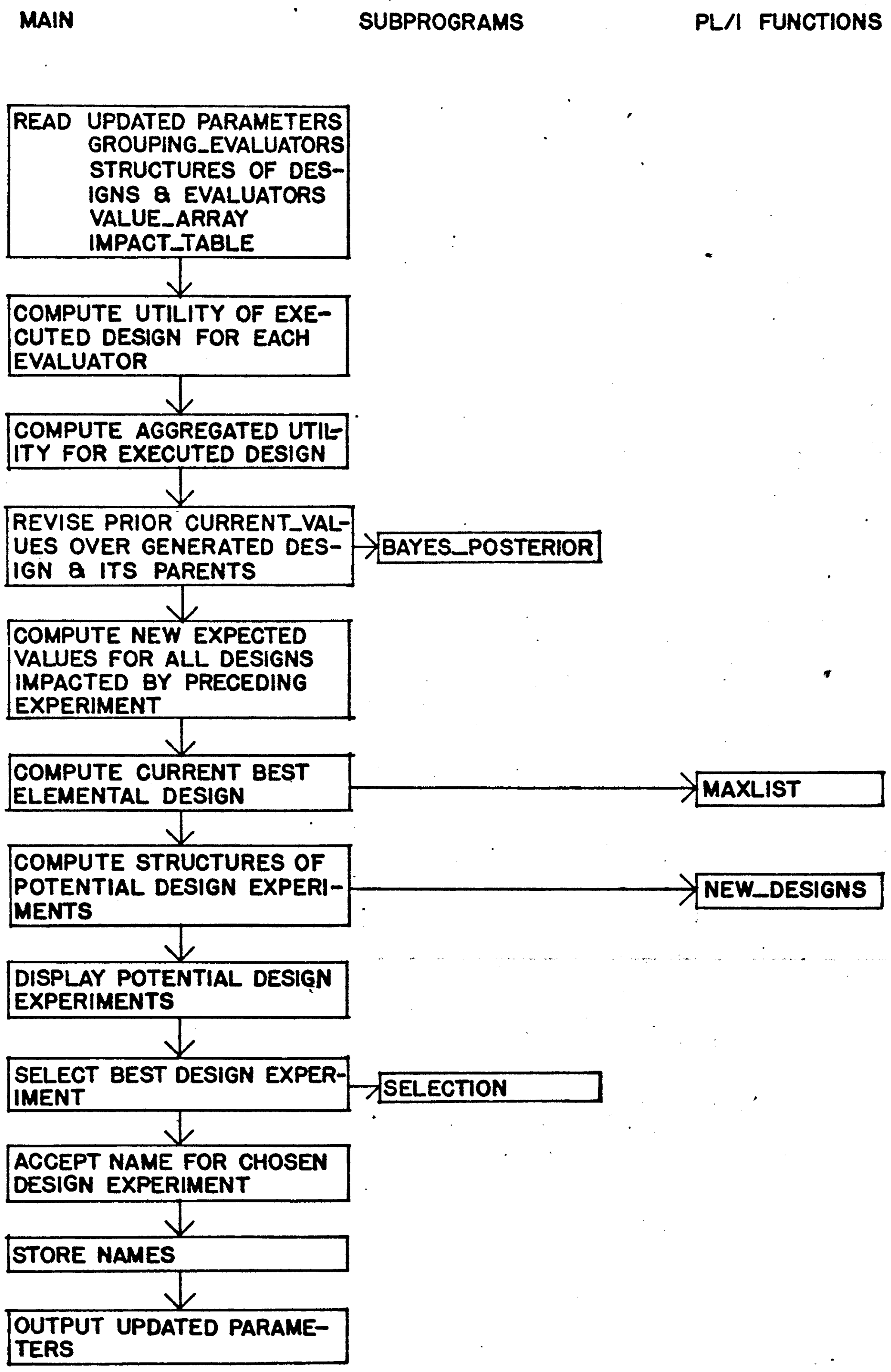

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
READ UPDATED PARAMETERS GROUPING_EVALUATORS STRUCTURES OF DESIGNS & EVALUATORS VALUE_ARRAY IMPACT_TABLE
COMPUTE UTILITY OF EXECUTED DESIGN FOR EACH EVALUATOR
COMPUTE AGGREGATED UTILITY FOR EXECUTED DESIGN
REVISE PRIOR CURRENT_VALUES OVER GENERATED DESIGN & ITS PARENTS
BAYES_POSTERIOR
COMPUTE NEW EXPECTED VALUES FOR ALL DESIGNS IMPACTED BY PRECEDING EXPERIMENT
COMPUTE CURRENT BEST ELEMENTAL DESIGN
MAXLIST
COMPUTE STRUCTURES OF POTENTIAL DESIGN EXPERIMENTS
NEW_DESIGNS
DISPLAY POTENTIAL DESIGN EXPERIMENTS
SELECT BEST DESIGN EXPERIMENT
SELECTION
ACCEPT NAME FOR CHOSEN DESIGN EXPERIMENT
STORE NAMES
OUTPUT UPDATED PARAMETERS

FIGURE NO. 4·28

BAYES_POSTERIOR (DISCOURSE)

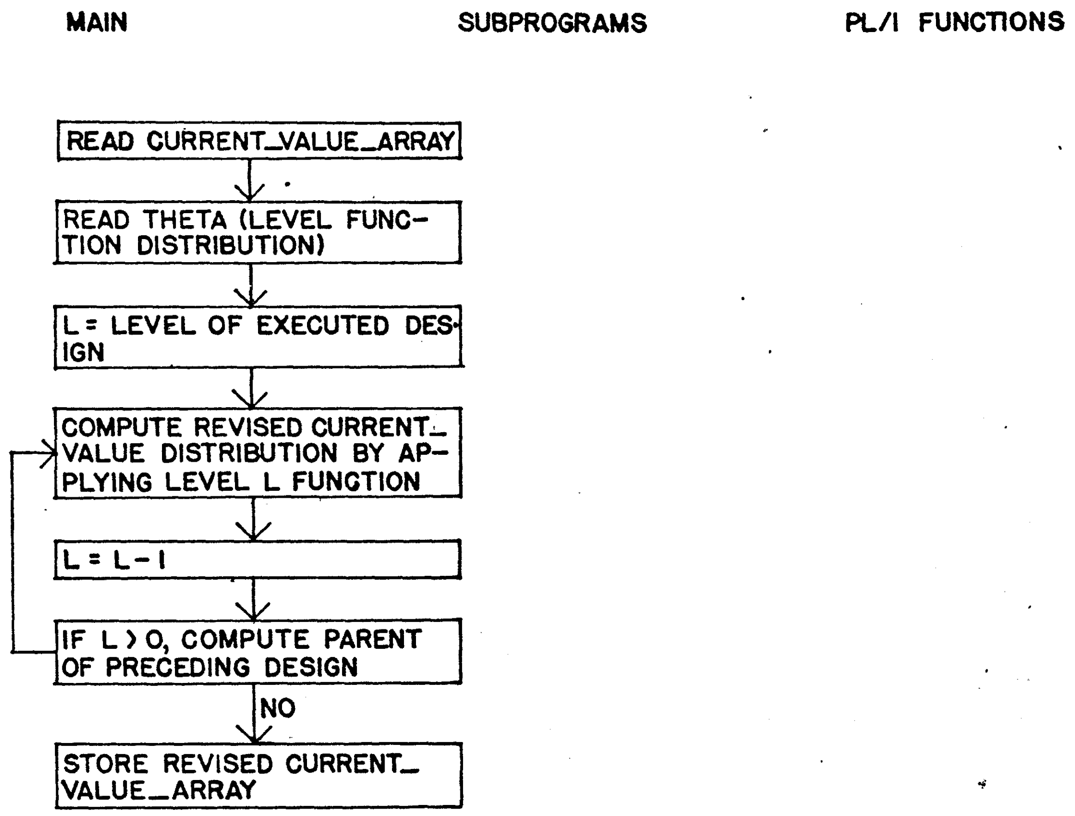

MAIN
SUBPROGRAMS
PL/I FUNCTIONS
READ CURRENT_VALUE_ARRAY
READ THETA (LEVEL FUNCTION DISTRIBUTION)
L = LEVEL OF EXECUTED DESIGN
COMPUTE REVISED CURRENT_VALUE DISTRIBUTION BY APPLYING LEVEL L FUNCTION
L = L − 1
IF L > 0, COMPUTE PARENT OF PRECEDING DESIGN
NO
STORE REVISED CURRENT_VALUE_ARRAY

FIGURE NO. 4·29

(d) computes expected current values of all designs impacted by the new design;

(e) determines the current best elemental design if one has been executed, otherwise states that no elemental design yet exists;

(f) continues as in Dynamic-Control-1, to compute the structure of potential design experiments (through PL.1 function "new designs");

(g) executes DISCOURSE subprogram "selection" to pick the best design experiment from the expected prior current values of their parents;

(h) accepts a name for the selected design, transfers it from potential to current status, and files updated parameters.

If at the completion of Dynamic Control, the designer decides that the expected improvement from the selected experiment does not meet his threshold criterion, then he does not generate the new design, the process stops, and he accepts the current results as output by the program.

The structure of the DISCOURSE sub-programs "selection" and "Bayes-Posterior" is fairly self-evident from their respective flow-charts. "selection" computes expected values for potential experiments and identifies the highest scoring possibility. "Bayes-Posterior" searches the Current design structure to derive the chain of inclusion from the new design, and updates the $f''_k(\theta/u(0,1,k))$ distributions for each impacted design, by Bayes' Theorem.

(e) Quasi-Additive Utility Aggregation

For two evaluators, x and y, the decision-maker in general, will have two one-dimensional utility functions; $u_x(x)$ for x, and $u_y(y)$ for y. These are related to the compound utility function by the scaling convention:

$$u(x_*,y_*) = u_x(x_*) = u_y(y_*) = 0;$$

$$u(x^*,y^*) = u_x(x^*) = u_y(y^*) = 1.0;$$

then, upon determination of $a_1$ and $a_2$, we have as above;

$$u(x,y) = a_1u_x(x) + a_2u_y(y) + (1 - a_1 - a_2)u_x(x)u_y(y).$$

For ease of computation, we could assume instead that the decision-maker has input the functions:

$$u(x_*,y) = a_2u_y(y)$$

$$u(x,y_*) = a_1u_x(x)$$

In this case the compound utility function becomes:

$$u(x,y) = u(x,y_*) + u(x_*,y) + ku(x,y_*)u(x_*,y)\text{, where}$$

$$k = \frac{1 - a_1 - a_2}{a_1a_2}.$$

For three scalar evaluators, x,y, and z, the assessment of the compound utility function is somewhat more complex. Given the three single-evaluator utility functions, $u_x(x)$, $u_y(y)$, and $u_z(z)$, related to the compound function by the scaling convention:

$$u(x_*,y_*,z_*) = u_x(x_*) = u_y(y_*) = u_z(z_*) = 0.0$$

$$u(x^*,y^*,z^*) = u_x(x^*) = u_y(y^*) = u_z(z^*) = 1.0,$$

upon the assessment of the 6 corner utilities $a_1$, $a_2$, $a_3$, $b_1$, $b_2$, and $b_3$, we derive:

$$\begin{aligned} u(x,y,z) = {} & a_1u_x(x) + a_2u_y(y) + a_3u_z(z) + (b_1-a_1-a_2)u_x(x)u_y(y) + \\ & (b_2-a_1-a_3)u_x(x)u_z(z) + (b_3-a_2-a_3)u_y(y)u_z(z) + \\ & (1-b_1-b_2-b_3+a_1+a_2+a_3)u_x(x)u_y(y)u_z(z) \end{aligned}$$

A geometrical interpretation is given in Figure No. 4.30. What must be assessed are the three utility functions represented by the heavy lines in the diagram, and the six circled corner utilities.

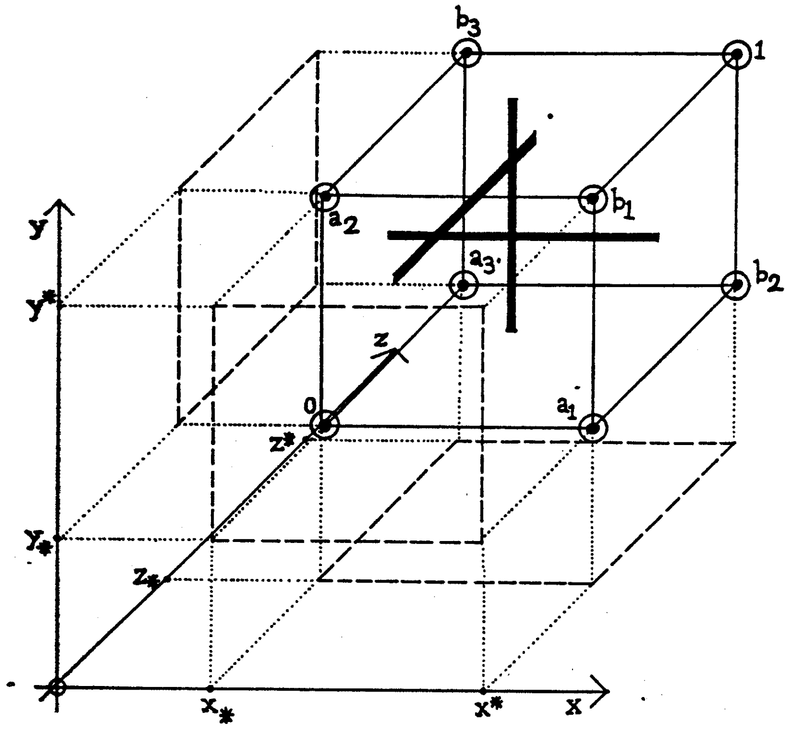


Figure No. 4.30

Again, for computational purposes, we could assume instead, that the decision-maker has the conditional utility functions:

$$u(x,y_*,z_*) = a_1u_x(x)$$

$$u(x_*,y,z_*) = a_2u_y(y)$$

$$u(x_*,y_*,z) = a_3u_z(z)$$

The resulting compound utility is:

$$u(x,y,z) = u(x,y_*,z_*) + u(x_*,y,z_*) + u(x_*,y_*,z) +$$

$$k_1u(x,y_*,z_*)u(x_*,y,z_*) + k_2u(x,y_*a_*)u(x_*,y_*,z) +$$

$$k_3u(x_*,y,z_*)u(x_*,y_*,z) + k_4u(x,y_*,z_*)u(x_*,y,z_*)u(x_*,y_*,z)$$

where:

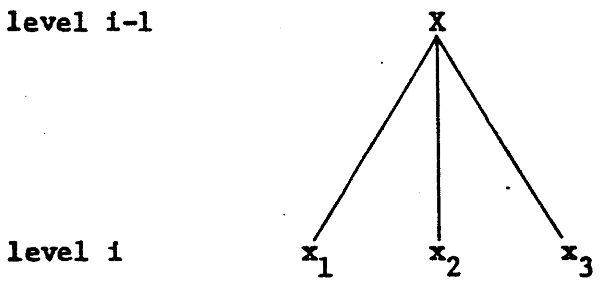
$$k_1 = \frac{b_1 - a_1 - a_2}{a_1 a_2}$$

$$k_2 = \frac{b_2 - a_1 - a_3}{a_1 a_3}$$

$$k_3 = \frac{b_3 - a_2 - a_3}{a_2 a_3}$$

$$k_4 = \frac{1 - b_1 - b_2 - b_3 + a_1 + a_2 + a_3}{a_1 a_2 a_3}$$

This form would be desirable if we were only aggregating utilities over a single level, say from level i to level i-1: e.g.



However, over a multi-level goal hierarchy, the aggregation of utilities from a lower level to the immediately higher level, results in single (vector) evaluator utilities, for arbitrary values of the other vector evaluators; rather than the conditional compound form. Therefore, we must work directly with the $a_1, \ldots a_n$

and $b_1, \ldots b_n$ corner utilities, rather than the pre-computed k factors. As an example, consider a simple 3 level goal tree, represented in Figure No. 4.31:

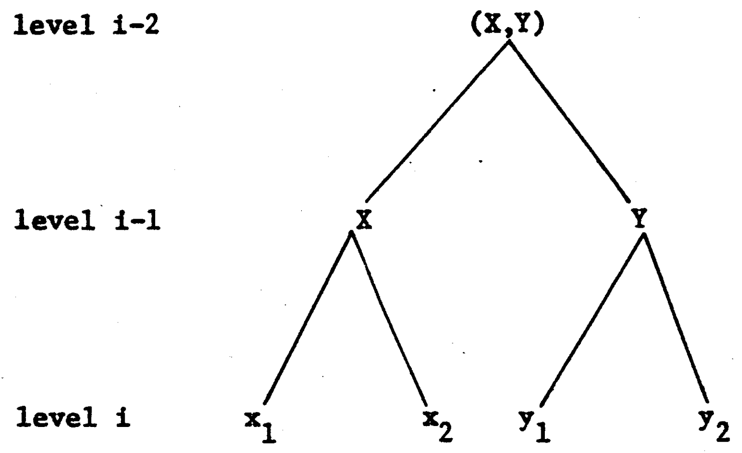


Figure No. 4.31

where $X = (x_1, x_2)$ and $Y = (y_1, y_2)$. We denote the first subscript as the level designation, and the second as group designation; aggregating from level i-1 to level i:

$$u_{i-1,X}(X) = a_{i,b} u_{i,x_1}(x_1) + a_{i,c} u_{i,x_2}(x_2) + (1 - a_{i,b} - a_{i,c}) u_{i,x_1}(x_1) u_{i,x_2}(x_2)$$

$$u_{i-1,Y}(Y) = a_{i,e} u_{i,y_1}(y_1) + a_{i,e} u_{i,y_2}(y_2) + (1 - a_{i,d} - a_{i,e}) u_{i,y_1}(y_1) u_{i,y_2}(y_2).$$

Aggregating from level i-1 to level i-2:

$$u_{i-2,XY}(X,Y) = a_{i-1,f}u_{i-1,X}(X) + a_{i-1,g}u_{i-1,Y}(Y) +$$

$$(1 - a_{i-1,f} - a_{i-1,g})u_{i-1,X}(X)u_{i-1,Y}(Y)$$

This procedure is followed in the computer programs, "Dynamic-Control", and "Quasi-Separable".

(f) Goal Decomposition

Given a set of n elemental goal variables or "evaluators",

$$G = (e_1, e_2, \ldots, e_n)$$

the goals hierarchy is structured by successively partitioning the set G into subsets of goal vectors which are mutually utility independent of one another; partitioning these subsets into further utility independent subsets, and so on, to the level of elemental evaluators. Each set of goal partitions defines a goal level. The partitioning may be done intuitively by the decision-maker for a small set of evaluators, or more structured decomposition algorithms such as Alexander's Hierarchical Decomposition (59) may be used. Decomposition via Alexander's method can serve to define the same levels for both evaluators and action descriptors (given our earlier discussion (*) which described it as a procedure which bridges both

(59) C. Alexander; op. cit.

(*) see page 72.

goal and planning domains). Figure No. 4.32 illustrates a possible decomposition for the 13 elemental evaluators of the North West Area Project:

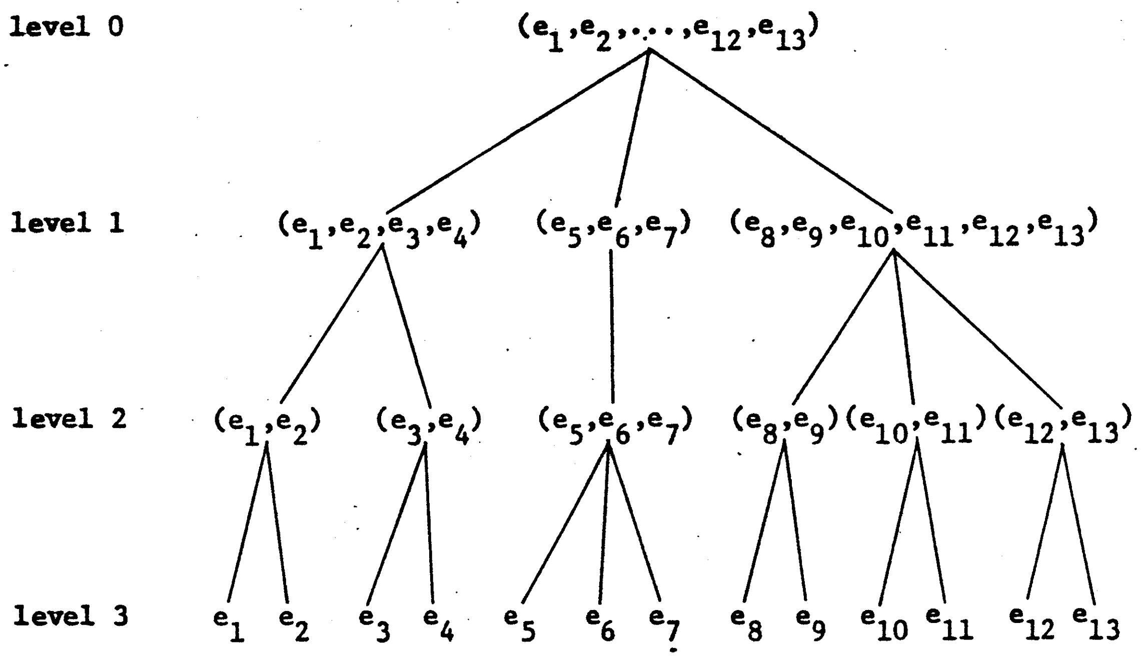

level 0
$(e_1, e_2, \ldots, e_{12}, e_{13})$
level 1
$(e_1, e_2, e_3, e_4)$
$(e_5, e_6, e_7)$
$(e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13})$
level 2
$(e_1, e_2)$
$(e_3, e_4)$
$(e_5, e_6, e_7)$
$(e_8, e_9)$
$(e_{10}, e_{11})$
$(e_{12}, e_{13})$
level 3
$e_1$ $e_2$ $e_3$ $e_4$ $e_5$ $e_6$ $e_7$ $e_8$ $e_9$ $e_{10}$ $e_{11}$ $e_{12}$ $e_{13}$

Figure No. 4.32

Evaluation at any level i proceeds by deriving utility assessments over individual level i goal vectors, aggregating in groups by the quasi-additive form to the next level i-1, and continuing to aggregate grouped utilities, level by level, until a single aggregated utility level 0 results. However, in practice, it may be:

(a) very difficult to get accurate assessments over, or measure vector evaluators, as opposed to scalar evaluators;

(b) less possible to measure evaluators (or even more important, to derive suitable evaluators) at upper levels, because of a less precise metric and fewer attributes vis-a-vis lower levels.

Therefore, we resort to an approximation which lends itself to the use of the hierarchically structured planning model, in that we assign probability distributions to intermediate level utilities, which reflect this lack of precision in goal measurement.

For each non-elemental, intermediate level i, we:

(1) determine the measurable evaluators or impacts which can be approximated by the level i metric;

(2) select, for each goal vector j, a principal component of that vector (or a weighted average of several components) from the measurable evaluators. Each selected evaluator serves as a surrogate for the level i goal vector of which it is a component. Surrogates approximate the real goals in the sense that they should induce behaviour consistent with, or as close as possible to, the real goals.

For example, in place of the level 1 evaluator, "maximize financial benefits", we select a principal component, such as "minimize overall project cost", as the surrogate, since it can be roughly measured at level 1. Figure No. 4.33 illustrates the North West Area Project goal structure again, with surrogate evaluators assigned to goal vectors:

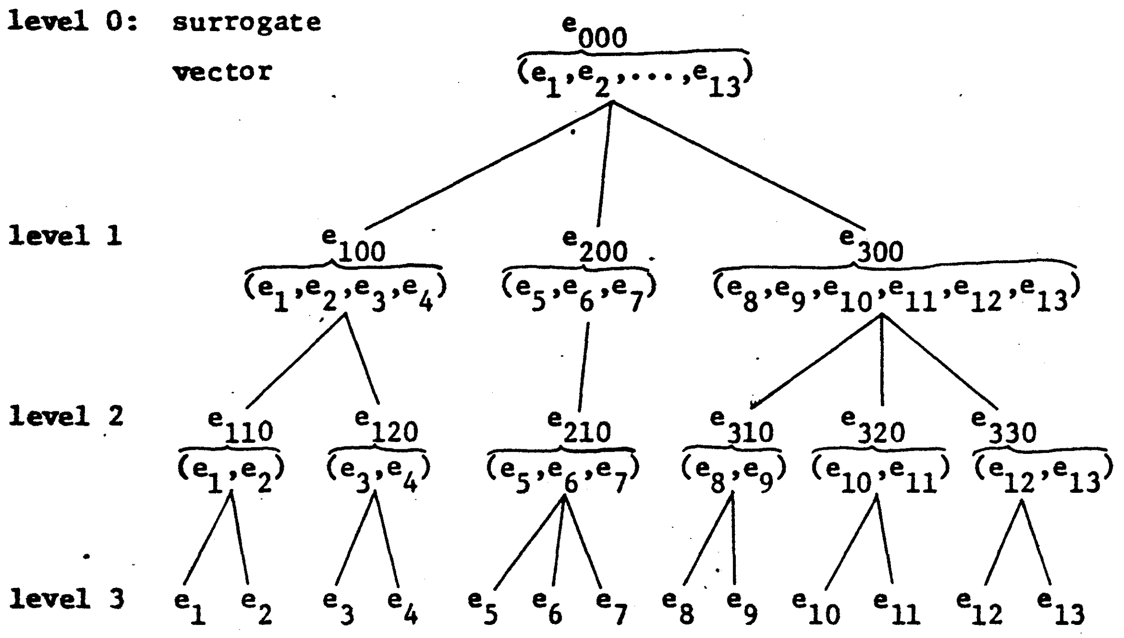


Figure No. 4.33

Level 3, as the elemental level, has no surrogate evaluators; "$e_{ooo}$", as the overall goal surrogate, is assigned the initial $f_0'(\theta)$ distribution, which is the decision-maker's assessment of the distribution of aggregated utilities resulting from the entire design process, and which starts off the entire dynamic search and evaluation process. As they are assessed throughout the process, the intermediate level i goal surrogates are also given probability distributions over their utility values, which reflect:

(a) the lack of precision at level i (a function of the scale of the metric);

(b) uncertainty as to how well the surrogate measure represents preferences for the entire goal vector.

The Bayesian posterior revision of the prior probability distributions over non-elemental utilities after each evaluation, allows the decision-maker to adjust the bias of the surrogate evaluator with respect to its lower level goal vector. It does not suggest if some other component in the goal vector would have been a better predictor. Of course, some goal variables may not be even roughly measurable at upper levels.

Two further points should be noted by way of explanation:

(1) we assume that utility functions can be assessed only for elemental evaluators, therefore, the utility measure for a surrogate evaluator is its elemental utility function. However, it is assumed, that the decision-maker can assess the corner utilities or tradeoffs among goal vectors at intermediate levels. (since this only involves combinations of the "best" and "worst" values of goals) Therefore, the aggregation of utilities from an intermediate goal level i, to level i-1, uses elemental level surrogate utility values, but combines these measures by means of corner utilities appropriate for level i. (However, aggregation from a lower level i+1, would yield vector utility values at level i, which are then combined with level i corner utilities for aggregation to level i-1).

(2) it is not required that evaluators be independent of one another (as required in the Fishburn additive utility model (60), for example) but rather than they can be combined into groups which are utility independent of each other. However, the resulting goal decomposition must be in a "planar tree" form, with no overlapping links.

(60) P. C. Fishburn; op. cit. (1965).

# 5. CONCLUSIONS AND EXTENSIONS

This study originally began with an investigation of the role of evaluation in the planning process as a terminal assessment procedure of design consequences with respect to explicit goal statements. It soon became apparent that restricting evaluation to this role also imposed an unnecessarily rigid conception of the problem-solving process on the planner or designer.

Firstly, the planner does not want to evaluate only full-developed alternatives at the end of the process, but may also wish to shortcut the planning process by gauging his progress at intermediate stages. Secondly, evaluation can play a useful interactive role in guiding the analyst towards better solutions and more efficient control of the planning process. The former consideration led to the inclusion of a set of independent routines which manipulate a basic impact matrix in various ways; the latter led to the incorporation of evaluation techniques within a hierarchical planning model. The three techniques: "User Operations", "Static Evaluation" and "Dynamic Evaluation" are separate and distinct entities in this paper; however, it should be stressed that ultimately they should be integrated into an overall evaluation "strategy", which would:

(1) judge the overall status of the planning process at any particular stage of execution;

(2) array the costs and benefits of various evaluation techniques (in terms of their contribution to the process), and suggest to the planner, which of these is most appropriate for his use at this stage. That is, the techniques would be evaluated as experiments in the Bayesian decision theory sense.

The usefulness of any evaluation technique is both project-independent and dependent: the costs and results of computation are relatively fixed, but the applicability to a problem context is closely tied to the project information (impacts, preferences, attributes, etc.) current at any time. Thus, benefits are highly variable from stage to stage within any particular planning process, as well as from project to project.

It may be of interest to the reader to compare the approach taken here with that of two other writers:

(1) John Boorn's CHOICE system for environmental design; (61)

(2) the capabilities in DODOTRANS, a computer language within the ICES System, for the evaluation of transportation systems, as exemplified by the work of John R. Mumford. (62)

---

(61) J. P. Boorn; A Choice System for Environmental Design and Development, (Cambridge, Mass., M.I.T. Dept. of Urban Studies and Planning, unpublished PhD. thesis, 1969).

(62) J. R. Mumford; Computer-Aided Evaluation of Transport Systems, (Cambridge, Mass., M.I.T. Dept. of Civil Engineering, Research Report R69-41, July, 1969).

Mumford's work implements a number of evaluation techniques, similar to those in "Static Evaluation":

(1) linear scoring function;

(2) utility theory(additive);

(3) cost-benefit analysis;

(4) goal fabric analysis. (63)

Operations allow the analyst to define goal hierarchies and evaluators, to evaluate, rank, and compare alternatives, generate new evaluators, and store the results in data files.

Boorn's thesis describes CHOICE, an evaluation system implemented in CTSS, and developed in conjunction with DISCOURSE. The user creates a system of evaluation accounts or matrices, on which various operations may be performed: arithmetic, definition of evaulators, computation of project costs and benefits, ordering, averages and standard deviations, discounting, scoring, etc. The routines implemented in CHOICE roughly correspond to the "user operations" described here.

Many of the basic operations described by both Boorn and Mumford did not have to be programmed explicitly here, because analogous capabilities already exist in DISCOURSE. (64) For

(63) Ibid.; p. 29.

(64) W. McMains et. al.; DISCOURSE Users' Manual, (Cambridge, Mass., M.I.T., 1971).

example, character string manipulation (for the naming of variables), file management and storage, arithmetic and logical operations, user interaction, etc. are all used implicitly in the present programs. Furthermore, the attribute data structure of DISCOURSE is adaptable to transformations of design attributes into consequences and evaluators; and the matrix operations required in evaluation, are readily programmed.

However, what may not be immediately apparent before actual use of a generalized evaluation system, is the immense amount of project- and user-specific information which must be prepared and input before the interactive capabilities of the system can be exploited. The time spent in using the evaluation techniques may only be a small percentage of the total time required for specification of the project in the system. In Mumford's work, the DODOTRANS system is tied to a highly specific set of prediction and analysis models and data for Northeast Corridor transportation planning, and although restricted to a relatively narrow class of problems, is also very operational on this account. Boorn develops a more generalized evaluation system, but since it is not related to a specific set of problems or model of the planning process, requires substantial data, prediction, and preference information, before it can be made operational for a specific project.

The approach here has tried to balance specific vs. generalized techniques. The component evaluation techniques are generalized only in the sense of illustrating a set of computations which would have to be adapted by a user for specific context. However, this meshes with the concept of DISCOURSE as a user-oriented computer language for urban design: the planner would develop his own models for the generation of designs and prediction of their consequences; and then adapt the techniques described here, for the assessment of the relative merits of alternatives. In contrast to both Boorn and Mumford, this paper has also tried to admit of more varied roles or problem-solving models, within which evaluation could function, in the planning process; and to link the concepts of multi-dimensionality and hierarchical problem structuring together in developing component routines.

Section 1, "INTRODUCTION", discussed a number of issues which point to possible extensions of the thesis:

(1) social choice: the elaboration of preference structures and choices for each significant actor group, and the display of impact matrices, comparisons, crucial trade-offs, points of agreement and disagreement among actors, etc. The system of accounts would serve as an information base for use in an negotiation and barganing process. Alternatively, a primary decision-maker may want to do a surrogate analysis in which he attempts to predict overall worth indices for alternatives, weighting actor preferences

by power and interest scores, or tradeoff measures. Many of the techniques for multi-dimensional evaluation are applicable in this latter case (i.e. interpreting actors as dimensions), however, results must be interpreted more cautiously, since surrogate aggregation is no substitute for true community interaction processes.

(2) User participation: the techniques here may be integrated into a comprehensive computer-based user interaction process, in which actors experiment with a number of alternative states (information bases) and vary their preferences and choices with respect to different consequence dimensions, and also through feedback from the preference of other actors. Evaluation techniques are applicable both to:

(a) gaming situations in which actors take on hypothetical, though reasonable roles and problem contexts;

(b) true negotiation situations, in which the information base and actor roles are relevant to an ongoing problem.

(3) Cost-benefit analysis and elaboration of preferences for time: routines may be added for computing Net Present Value, Internal Rate of Return, Benefit-Cost Ratios, etc. Multidimensional utility theory may also be used if standard

discounting formulae are not suitable to express the decision-maker's preference structure. Such capabilities were not implemented in the current set of programs, partly because of the concentration on multi-dimensionality across monetary and non-monetary consequences, and partly because the various discounting formulae give contradictory criteria for choice even among projects with only monetary consequences.

(4) self-organization: this refers to non-arbitrary ways of introducing new "images" of the problem within the planning process: deriving new preference structures, changing dimensions of the evaluation or search spaces, guidance of the search effort towards sub-optimal results, control and allocation of analysis resources among the different problem-solving activities. Extensions in this area involve expanding the more conventional notions of evaluation within problem-solving paradigms such as systems analysis or decision theory, towards research in artificial intelligence, as suggested, for example in Minsky's article, quoted above. (65)

The work here could also be usefully complemented by an empirically-based descriptive study, which attempts to outline the difficulties in deriving preference information from actors

(65) M. Minsky; op. cit.

in complex urban planning problems, in aggregating these preferences over a number of consequence dimensions, and in applying the models described in on-going planning and design processes. By taking a more theoretical perspective of evaluation techniques, we may have partially avoided the inevitable confrontation which must accompany the transition of these ideas to their implementation in the real world.

## 6. BIBLIOGRAPHY

(1) Ackoff, R. L.; Scientific Method: Optimizing Applied Research Decisions, (New York, N. Y., John Wiley & Sons, Inc., 1962)

(2) Alexander, C.; Notes on the Synthesis of Form, (Cambridge, Mass., Harvard University Press, 1964)

(3) Boorn, J. P.; A Choice System for Environmental Design and Development, (Cambridge, Mass., M.I.T. Dept. of Urban Studies & Planning, unpublished PhD. thesis, 1969)

(4) Coombs, C. H.; Raiffa, H.; Thrall, R. M.; "Some Views on Mathematical Models and Measurement Theory", in Thrall, Coombs, & Davis, eds.; Decision Process, (New York, N. Y., John Wiley & Sons, Inc., 1954), pp. 19 - 37

(5) Corporation Joint Advisory Committee on Institute-Wide Affairs; Report on Simplex and Related Development, (Cambridge, Mass., M.I.T., 5 June, 1970)

(6) Emery, J. C.; Organizational Planning and Control Systems: Theory and Technology, (New York, N. Y., MacMillan Co., 1969)

(7) Ernst, G. W.; "GPS and Decision Making: An Overview, in Banerji, R.; & Mesarovic, M.D.; eds.; Theoretical Approaches to Non-Numerical Problem-Solving, (New York, N. Y., Springer-Verlag, 1970)

(8) Fishburn, P. C.; Decision and Value Theory, (New York, N. Y., John Wiley & Sons, Inc., 1964)

(9) -----; "Independence in Utility Theory with Whole Product Sets", Operations Research, (Vol. 13, 1965), pp. 28 - 45

(10) Harary, F.; Norma, R.Z.; Cartwright, D.; Structural Models: An Introduction to the Theory of Directed Graphs, (New York, N. Y., John Wiley & Sons, Inc., 1965)

(11) Johnsen, E.; Studies in Multi-Objective Decision Models, (Lund, Studentlitteratur, Economic Research Center in Lund, Monograph No. 1, 1968)

(12) Johnson, H.W.; public announcement re Simplex purchase, (Cambridge, Mass., 10 July, 1969)

(13) Keeney, R.L.: Multidimensional Utility Functions: Theory, Assessment and Application, (Cambridge, Mass., M.I.T. Operations Research Center, Technical Report No. 43, Oct., 1969)

(14) Maass, A.; & Hufschmidt, M.; Design of Water Resource Systems, (Cambridge, Mass., Harvard University Press, 1962)

(15) MacCrimmon, K.R.: Improving the System Design and Evaluation Process by the Use of Trade-off Information: An Application to Northeast Corridor Transportation Planning, (New York, N. Y., Rand Corporation memo RM-5877-DOT, 1969)

(16) Manheim, M. L.; Hierarchical Structure: A Model of Design and Planning Processes, (Cambridge, Mass., M.I.T. Press, 1966)

(17) ----- & Hall, F.; Abstract Representation of Goals, (Cambridge Mass., M.I.T. Dept. of Civil Engineering, Professional Paper P67-24, January, 1968)

(18) ----- et. al.; The Impacts of Highways upon Environmental Values, (Cambridge, Mass., M.I.T. Urban Systems Laboratory, Report No. USL-69-1, March, 1969)

(19) McMains, W.; et. al.; DISCOURSE User's Manual, (Cambridge, Mass., M.I.T., 1971)

(20) Mesarovic, M.D.: Macko, D.; Takahara, Y.; Theory of Hierarchical Multilevel Systems, (New York, N. Y., Academic Press, 1970)

(21) Miller, G.A.; Galanter, E.; & Pribram, K.H.: Plans and the Structure of Behaviour, (New York, N.Y., Holt, Rinehart & Wilson, Inc., 1960)

(22) Miller, J.R., III; The Assessment of Worth: A Systematic Procedure and Its Experimental Validation, (Cambridge, Mass., M.I.T. Sloan School of Management, unpublished PhD. thesis, 1966)

(23) Minsky, M.A.; "Steps Toward Artificial Intelligence", in Feigenbaum, E.A.; & Feldman, J.; eds.; Computers and Thought, (New York, N. Y., McGraw-Hill, 1963)

(24) Mumford, J.R.: Computer-Aided Evaluation of Transport Systems, (Cambridge, Mass., M.I.T. Dept. of Civil Engineering, Research Report R69-41, 1969)

(25) Neft, D.S.; Statistical Analysis for Areal Distributions, (Philadelphia, Pa., Regional Science Institute, Monograph series no. 2, 1966)

(26) von Neumann, J.; & Morgenstern, O.; Theory of Games and Economic Behaviour, (Princeton, N.J., Princeton University Press, 1947)

(27) Newell, A.; Shaw, J.C.; & Simon, H.A.; "A general problem-solving program for a computer", Computers and Automation, (Vol. 8, No. 7, 1959), pp. 10 - 16

(28) Pecknold, W.M.; The Evolution of Transport Systems: An Analysis of Time-Staged Investment Strategies Under Uncertainty, (Cambridge, Mass., M.I.T. Dept. of Civil Engineering, unpublished PhD. thesis, 1970)

(29) Raiffa, H.; Tradeoffs Under Certainty, (Cambridge, Mass., Harvard University, unpublished notes, 1968)

(30) Reitman, W.R.; "Heuristic decision procedures, open constraints, and the structure of ill-defined problems", in Shelley, M.W., III; & Bryan, G.L.: eds.; Human Judgments and Optimality, (New York, N.Y., John Wiley & Sons, Inc., 1964), Ch. 15, pp. 282 - 315

(31) Simon, H.A.; "A Behavioural Model of Rational Choice", in -----; Models of Man, (New York, N.Y., John Wiley & Sons, Inc., 1957)

(32) -----; The Sciences of the Artificial, (Cambridge, Mass., M.I.T. Press, 1969)

(33) Simplex Advisory Committee; Considerations in the Future Development of Simplex and Related M.I.T. Properties, (Cambridge, Mass., M.I.T., Feb., 1970)

(34) Torgerson, W.S.; Theory and Methods of Scaling, (New York, N.Y.; John Wiley & Sons, Inc., 1958)

# 7. APPENDIX

## 7.1 DISCOURSE Program Listings

### (a) User Operations

Evaluators

Preliminary

order

display-impacts

Pareto

compare

display-transform

satisfaction

### (b) Static Evaluation

Single-Pass

Fishburn-Relative-Value

Case-Relative-Value

Linear-Scoring

Quasi-Separable

### (c) Dynamic Evaluation

Dynamic-Control-1

Dynamic-Control

selection

Bayes-Posterior

(d) <u>PL/1 Functions</u>

ordering.pl 1

maxlist.pl 1

minlist.pl 1

quasi-order.pl 1

new-designs.pl 1

Evaluators

```
dfa level1 1: (1,25)
dfa name_evaluators 2: (1,25),(0,2)
dfa structure_goals 2: (1,25),(1,4)
expand "//"
say Type in the number of evaluator names to be input
expand "/"
read_set console
read: $1num_evaluators=
expand "//"
say Type in the name of each evaluator (maximum 10 characters) in
say order, preceded by a single digit level number. For example:
say  :0 Maxbenefit     (overall goal)
say  :1 Financial      (1st secondary objective, under overall goal)
say  :2 Cambridge      (1st tertiary objective, under "Financial")
say  :3 tax_yields     (1st lower-level goal, under "Cambridge")
say  :3 serv_costs     (2nd lower-level goal, under "Cambridge")
say  :2 MIT            (2nd tertiary objective, under "Financial")
expand "//"
through NE1, for i= 1. until num_evaluators
read: $1level1(i) $2name_evaluators(i,0)
NE1$ continue
through NE2, for j= 2. until num_evaluators
k=j-1.
level2 = level1(j)
if (level2.leq.1.0) goto NE6
end = level2-1.
through NE3, for l= 1. until end
structure_goals(j,l) = structure_goals(k,l)
NE3$ continue
NE6$ structure_goals(j,level2) = structure_goals(k,level2)+1.
NE2$ continue
sa structure structure_goals
expand "/array structure_goals stored//"
sa names name_evaluators
expand "array name_evaluators stored//"
read_console_return
dfa group 2: (1,3),(1,10)
through NE4, for m = 1. until 3.
j = 1.
through NE5, for i = 2. until num_evaluators
if (structure_goals(i,m).eql.0.) goto NE6
if (structure_goals(i,(m+1)).neq.0.) goto NE5
group(m,j) = group(m,j) + 1.
goto NE5
NE6$ if (structure_goals((i-1),m).eql.0.) goto NE5
j = j+1
NE5$ continue
NE4$ continue
sa grouping_evaluators group
say array "group" stored
read_console
```

Preliminary

```
expand "/Type in the number of evaluators//"
read_set console
read: $1num_evaluators=
expand "//"
expand "Type in the number of designs//"
read: $1current_designs=
expand "//"
dfa impact 2: (2,25).(1,20)
rs impacts
through X1, for i = 2, until num_evaluators
read: $1impact(i,2)...impact(i,current_designs)
X1$ continue
rs off
expand "Filename impacts read//"
ex names
expand "Filename names read//"
dfa rank 2: (1,25),(1,20)
dfa temp 2: (1,5),(0,2)
zoink = 0.
say Programs accept single-level or hierarchically-structured
say evaluation problems:
say Type 1 if designs and goals both have only one level of
say generality; the number of hierarchical levels otherwise.
expand "/"
read_set console
read: $1zoink =
if (zoink.eql.1.) goto P7
dfa e1 1: (1,15)
dfa d1 1: (1,15)
expand "/"
expand "Type in the level number desired: 1, 2, or 3//"
read: $1level=
expand "//"
ex structure
expand "Filename structure read//"
m = 0.
n = 0.
through P5, for i = 2, until num_evaluators
if ((structure_goals(i,level).eql.0.).or.(structure_goals(i,(level+1))&
.neq.0.)) goto P5
m = m+1.
e1(m) = i
P5$ continue
m1 = m
comment: m1 is the number of evaluators at the chosen level
through P6, for j = 1, until current_designs
if ((structure_designs(j,level).eql.0.).or.(structure_designs(j,&
(level+1)).neq.0.)) goto P6
n = n+1.
d1(n) = j
P6$ continue
n1 = n
```

```
comment: n1 is the number of designs at the chosen level
goto P10
P7$ m1 = num_evaluators
n1 = current_designs
through P8, for i = 1, until m1
e1(i) = i
P8$ continue
through P9, for j = 1, until n1
d1(j) = j
P9$ continue
P10$ return
```

order

```
00$ dfa temp_impact 1: (1,20)
dff ordering (,20,rank,0,1)
expand "/"
say Type in the name of the evaluator to be used
expand "/"
read_set console
read: $2temp(1,0)
expand "//"
say Type in "increasing" for ranking by increasing value, or
say type in "decreasing" for ranking by decreasing value.
expand "/"
read: $2temp(2,0)
through 01, for i = 1. until m1
x = e1(i)
if ceql_F(name_evaluators(x,0),temp(1,0)) goto 02
01$ continue
02$ n = x
if ceql_F(temp(2,0),"decreasing") m = 1.
if ceql_F(temp(2,0),"increasing") m = 0.
through 03, for j = 1. until n1
y = d1(j)
temp_impact(j) = impact(n,y)
03$ continue
comment:
call ordering(temp_impact,n1,rank,m,1)
comment:
set_field_width 12
set_carriage_width 72
set_decimal_places 0
expand "///Designs           Ranking///"
through 04, for k = 1. until n1
y = d1(k)
expand "$1name_designs(y,0)*   $3rank(1,k)//"
04$ continue
expand "///"
read_console
```

display_impacts

```
expand "/"
say For display, choose 2 to 5 designs from the following list:
through D1, for i = 1, until n1
y = d1(i)
expand "  $3y:  $1name_designs(y,0)/"
D1$ continue
expand "//Type in the number of designs to be displayed://"
read_set console
read: $1num_designs=
expand "/Type in the names of the designs//"
through D2, for j = 1, until num_designs
read: $2temp(j,0)
D2$ continue
expand "///"
comment:
set_field_width 12
set_decimal_places 2
expand "Designs:          "
dfa temp1 1: (1,5)
through D3, for k = 1, until num_designs
through D4, for l = 1, until n1
y = d1(l)
if ceql_f(temp(k,0),name_designs(y,0)) goto D5
D4$ continue
D5$ temp1(k) = d1(l)
expand "$1name_designs(y,0)*  "
D3$ continue
expand "//Impacts//"
through D6, for i = 1, until m1
x = e1(i)
expand "$1name_evaluators(x,0)*  "
through D7, for j = 1, until num_designs
y = temp1(j)
expand "$1impact(x,y)*"
D7$ continue
expand "//"
D6$ continue
expand "//"
read_console
```

Pareto

```
dfa sum 2:(1,20),(1,20)
dfa reach 2:(1,20),(1,20)
dfa temp_impact 2: (1,15),(1,15)
dff ordering(,20,rank,0,1)
dff quasi_order(rank,sum,reach,m1,n1)
dff maxlist(,30,0,1)
expand "/Construction of Ordinal Ranking Matrix//"
say Type in "increasing" for ranking by increasing value, or
say type in "decreasing" for ranking by decreasing value, for
say each evaluator:
expand "//"
t = 0.
through P1, for i = 1. until m1
x = e1(i)
through P2, for j = 1. until n1
y = d1(j)
temp_impact(i,j) = impact(x,y)
P2$ continue
P1$ continue
read_set console
through P3, for i = 1. until m1
x = e1(i)
expand "$1name_evaluators(x,0)/"
read: $2temp(1,0)
expand "//"
if ceql_F(temp(1,0),"decreasing") t = 1.
if ceql_F(temp(1,0),"increasing") t = 0.
comment:
call ordering(temp_impact(i,1),n1,rank(i,1),t,1)
comment:
P3$ continue
expand "/Ordinal Ranking Matrix completed///"
say Dominance check by constructing quasi-levels
comment:
call quasi_order(rank,sum,reach,m1,n1)
comment:
m = 0.
call ordering(sum,n1,rank,m,1)
comment
set_decimal_places 0
set_carriage_width 72
set_field_width 0
expand "//Quasi-level 1: Pareto-efficient frontier//"
through P4, for i = 1. until n1
if rank(1,i).neq.1.) goto P4
y = d1(i)
expand "$1name_designs(y,0)/"
P4$ continue
expand "///"
val = 0.
index = 1.
comment:
```

```
call maxlist(rank(1,1),n1,val,index)
comment: val gives the number of quasi-levels produced
if (val.eql.1.) goto end
through P5, for i = 2, until val
expand "Quasi-level $3i:  Dominated alternatives//"
through P6, for j = 1, until n1
if (rank(1,j).neq.i) goto P6
y = d1(j)
expand "$1name_designs(y,0)/"
P6$ continue
expand "///"
P5$ continue
end$ read_console
```

compare

```
dfa difference 1: (1,15)
expand "/"
say Type in the names of the two designs to be compared
expand "/"
read_set console
through C1, for j = 1, until 2,
read: $2temp(j,0)
C1$ continue
through C2, for k = 1, until n1
y = d1(k)
if ceql_F(name_designs(y,0),temp(1,0)) goto C3
C2$ continue
C3$ n1 = y
through C4, for l = 1, until n1
y = d1(l)
if ceql_F(name_designs(y,0),temp(2,0)) goto C5
C4$ continue
C5$ n2 = y
through C6, for i = 1, until m1
x = e1(i)
difference(i) = impact(x,n1) - impact(x,n2)
C6$ continue
set_carriage_width 72
set_field_width 12
set_decimal_places 2
expand "                  $1name_designs(n1,0)*"
expand "  $1name_designs(n2,0)*"
expand "Difference///"
expand "Evaluators  //"
through C7, for j = 1, until m1
x = e1(j)
expand "$1name_evaluators(x,0)*  $1impact(x,n1)*"
expand "$1impact(x,n2)*  $1difference(j)//"
C7$ continue
expand "//"
read_console
```

display_transform

```
dfa value 2: (2,25),(1,24)
rs value_array
through DT1, for i = 2. until num_evaluators
read: $1value(i,1)...value(i,24)
DT1$ continue
expand "/Filename value_array read//"
rs off
dfa transform 2: (2,25),(1,20)
through DT2, for j = 1. until m1
x = e1(j)
through DT3, for k = 1. until n1
y = d1(k)
z = impact(x,y)
transform(x,y) = value(x,z)
DT3$ continue
DT2$ continue
comment:
sa transform_array transform
say For display, choose 2 to 5 designs from the following list:
expand "/"
through DT4, for i = 1. until n1
y = d1(i)
expand "  $3y:  $1name_designs(y,0)/"
DT4$ continue
expand "//"
comment:
expand "Type in the number of designs to be displayed//"
read_set console
read: $1num_designs=
expand "/Type in the names of the designs//"
through DT5, for j = 1. until num_designs
read: $2temp(j,0)
DT5$ continue
expand "///"
comment:
dfa temp1 1: (1,5)
set_field_width 12
set_decimal_places 2
expand "Designs:            "
through DT6, for k = 1. until num_designs
through DT7, for l = 1. until n1
y = d1(l)
if ceql_F(temp(k,0),name_designs(y,0)) goto DT8
DT7$ continue
DT8$ temp1(k) = d1(l)
expand "$1name_designs(y,0)*  "
DT6$ continue
expand "//Transforms//"
through DT9, for i = 1. until m1
x = e1(i)
expand "$1name_evaluators(x,0)*  "
through DT10, for j = 1. until num_designs
```

```
k = temp1(j)
expand "$1transform(x,k)*"
DT10$ continue
expand "//"
DT9$ continue
read_console
```

satisfaction

```
S0$ dff ordering(,20,rank,0,1)
expand "/"
say Type in the name of the design to be analysed
expand "/"
read_set console
read: $2temp(1,0)
through S1, for j = 1. until n1
y = d1(j)
if ceql_F(temp(1,0),name_designs(y,0)) goto S2
S1$ continue
S2$ n = y
dfa temp_goals 1: (1,20)
through S4, for j = 1. until m1
x = e1(j)
temp_goals(j) = transform(x,n)
S4$ continue
comment:
f = 0.
g = 1.
call ordering(temp_goals,m1,rank,f,g)
comment:
set_field_width 12
set_carriage_width 72
set_decimal_places 2
expand "//Design:    $1temp(1,0)///"
expand "Evaluator        Utility          Rank//"
through S5, for i = 1. until m1
x = e1(i)
expand "$1name_evaluators(x,0)* $1temp_goals(i)*  "
expand "$3rank(1,i)//"
S5$ continue
expand "//"
read_console
```

Single_Pass

```
dfa reach 2:(1,20),(1,20)
dfa sum 2:(1,20),(1,20)
dfa temp_impact 2: (1,15),(1,15)
dff ordering(,30,rank,m1,n1)
through SP1, for i = 1. until m1
x = e1(i)
through SP2, for j = 1. until n1
y = d1(j)
temp_impact(i,j) = impact(x,y)
SP2$ continue
SP1$ continue
comment: construct Ordinal Ranking Matrix
expand "//"
say Type in "increasing" for ranking by increasing value, or
say type in "decreasing" for ranking by decreasing value; for
say each evaluator
expand "//"
t = 0.
through SP8, for i = 1. until m1
x = e1(i)
expand "$1name_evaluators(x,0)/"
read_set console
read; $2temp(1,0)
expand "//"
if ceql_F(temp(1,0),"decreasing") t=1.
if ceql_F(temp(1,0),"increasing") t=0.
comment:
call ordering(temp_impact(i,1),n1,rank(i,1),t,1)
comment:
SP8$ continue
expand "//Ordinal ranking matrix completed//"
say Dominance check by constructing quasi-levels
dff quasi_order(rank,sum,reach,m1,n1)
comment:
call quasi_order(rank,sum,reach,m1,n1)
comment:
call ordering(sum,n1,rank,0,1)
comment:
set_decimal_places 0
set_carriage_width 72
set_field_width 0
expand "//Quasi-level 1: Pareto-efficient frontier//"
through SP9, for i = 1. until n1
if (rank(1,i).neq.1.) goto SP9
x = d1(i)
expand "$1name_designs(x,0)/"
SP9$ continue
expand "///"
val = 0.
index = 1.
dff maxlist(,30,0,1)
comment:
```

```
call maxlist(rank(1,1),n1,val,index)
comment: val gives the number of quasi-levels produced
if (val.eql.1.) goto SP14
through SP10, for i = 2, until val
expand "Quasi-level $3i: Dominated alternatives//"
through SP11, for j = 1, until n1
if (rank(1,j).neq.i) goto SP11
x = d1(j)
expand "$1name_designs(x,0)/"
SP11$ continue
expand "///"
SP10$ continue
dfa reduced_impact 2: (1,15),(1,15)
comment: construction of reduced impact matrix
comment: (this section is by-passed if there are no dominated altern-
comment: atives)
n = 0.
through SP12, for i = 1, until n1
if (rank(1,i).neq.1.) goto SP12
n = n+1.
d1(n) = d1(i)
through SP13, for j = 1, until m1
reduced_impact(j,n) = temp_impact(j,i)
SP13$ continue
SP12$ continue
n1 = n
comment: reduced impact matrix completed
goto SP17
SP14$ dff move(,,)
dfa reduced_impact 2:(1,15),(1,15)
comment:
call move(temp_impact,reduced_impact)
comment:
expand "/temp_impact copied into reduced impact//"
SP17$ read_console return
dfa value 2: (2,25),(1,24)
rs value_array
through SP18, for i = 2, until num_evaluators
read: $1value(i,1)...value(i,24)
SP18$ continue
rs off
expand "/Filename value_array read/"
say Choose evaluation method by typing in one of the following
say names:
say   (1) Fishburn Relative Value
say   (2) Case Relative Value
say   (3) Linear Scoring (also Independent Utility)
say   (4) Quasi-Separable Utility
expand "/"
SP19$ read_set console
read: $2temp(2,0)
if ceql_F(temp(2,0),"Fishburn") ex Fishburn_Relative_Value
if ceql_F(temp(2,0),"Case") ex Case_Relative_Value
if ceql_F(temp(2,0),"Linear") ex Linear_Scoring
if ceql_F(temp(2,0),"Quasi-Separ") ex Quasi_Separable
say spelling mistake: type name again
goto SP19
read_console
```

Fishburn_Relative_Value

```
set_decimal_places 2
dfa weight 1: (1,15)
dfa standard 2: (1,15),(1,15)
dfa Fishburn_rv 2: (1,15),(1,15)
dfa total_frv 1: (1,15)
ex transform_array
expand "//Filename transform_array read//"
dff maxlist(,30,0,1)
dff minlist(,30,1000,1)
comment:
dfa t1 2:(1,15),(1,15)
through FRV1, for i = 1, until m1
x = e1(i)
through FRV2, for j = 1, until n1
y = d1(j)
t1(i,j) = transform(x,y)
FRV2$ continue
FRV1$ continue
val = 0,
index = 1,
sum_weight = 0,
through FRV3, for i = 1, until m1
x = e1(i)
call maxlist(t1(i,1),n1,val,index)
max = val
call minlist(t1(i,1),n1,val,index)
min = val
through FRV4, for j = 1, until n1
y = d1(j)
standard(i,j) = (t1(i,j)-min)/(max-min)
FRV4$ continue
weight(i) = max-min
sum_weight = sum_weight + weight(i)
FRV3$ continue
through FRV, for i = 1, until m1
weight(i) = weight(i)/sum_weight
FRV$ continue
comment:
through FRV5, for l = 1, until n1
through FRV6, for k = 1, until m1
Fishburn_rv(k,l) = weight(k)*standard(k,l)
total_frv(l) = total_frv(l) + Fishburn_rv(k,l)
FRV6$ continue
FRV5$ continue
dfa temp1 1: (1,5)
expand "//"
set_field_width 12
say For display, choose 2 to 5 designs from the following list:
expand "/"
through FRV7, for i = 1, until n1
y = d1(i)
expand "$3y:  $1name designs(y,0)/"
```

```
FRV7$ continue
expand "/"
comment:
say Type in the number of designs to be displayed:
expand "/"
read_set console
read: $1num_designs=
expand "/"
say Type in the names of the designs:
expand "/"
through FRV8, for j = 1, until num_designs
read: $2temp(j,0)
FRV8$ continue
expand "/"
comment:
set_decimal_places 2
expand "Designs:       "
through FRV9, for k = 1, until num_designs
through FRV10, for l = 1, until n1
x = d1(l)
if ceql_F(temp(k,0),name_designs(x,0)) goto FRV11
FRV10$ continue
FRV11$ temp1(k) = l
expand "  $1name_designs(x,0)*"
FRV9$ continue
expand "//Weighted/Values//"
through FRV12, for i = 1, until m1
y = e1(i)
expand "$1name_evaluators(y,0)"
through FRV13, for j = 1, until num_designs
k = temp1(j)
expand "$1Fishburn_rv(i,k)*"
FRV13$ continue
expand "//"
FRV12$ continue
expand "Weighted/Total      "
through FRV14, for k = 1, until num_designs
l = temp1(k)
expand "$1total_frv(l)*"
FRV14$ continue
expand "//"
comment:
f = 0,
g = 1,
call ordering(total_frv,n1,rank,f,g)
comment:
expand "Ranking:  "
through FRV15, for i = 1, until num_designs
x = temp1(i)
expand "$3rank(1,x)*"
FRV15$ continue
expand "//"
read_console
```

Case_Relative_Value

```
set_decimal_places 4
dfa Case_rv 2: (1,15),(1,15)
dfa total_crv 1: (1,15)
dfa weight 1:(2,25)
rs weight_table
read: $1weight(2)...weight(num_evaluators)
rs off
expand "/Filename weight_table read//"
ex transform_array
expand "Filename transform_array read//"
dfa t1 2:(1,15),(1,15)
through CRV1, for i = 1, until m1
x = e1(i)
through CRV2, for j = 1, until n1
y = d1(j)
if (transform(x,y).eql.0.) goto CRV
t1(i,j) = weight(x)/transform(x,y)
goto CRV2
CRV$ t1(i,j) = 1000.
CRV2$ continue
CRV1$ continue
comment:
through CRV3, for i = 1, until m1
denominator = 1.
through CRV4, for j = 2, until n1
denominator = denominator + t1(i,1)/t1(i,j)
CRV4$ continue
Case_rv(i,1) = (1./denominator)
through CRV5, for k = 2, until n1
Case_rv(i,k) = (t1(i,1)/t1(i,k))*Case_rv(i,1)
CRV5$ continue
CRV3$ continue
through CRVV, for j = 1, until n1
through CRVW, for i = 1, until m1
x = e1(i)
total_crv(j) = total crv(j) + weight(x)*case_rv(i,j)
CRVW$ continue
CRVV$ continue
comment:
dfa temp1 1: (1,5)
set_field_width 12
say For display, choose 2 to 5 designs from the following list:
expand "/"
through CRV6, for i = 1, until n1
y = d1(i)
expand "$3y:  $1name designs(y,0)/"
cRV6$ continue
expand "/"
comment:
say Type in the number of designs to be displayed:
expand "/"
read_set console
```

```
read: $1num_designs=
expand "/"
say Type in the names of the designs to be displayed!
expand "/"
through CRV7, for j = 1, until num_designs
read: $2temp(j,0)
CRV7$ continue
expand "/"
comment:
expand "Designs:        "
through CRV8, for k = 1, until num_designs
through CRV9, for l = 1, until n1
x =d1(l)
if ceql_F(temp(k,0),name_designs(x,0)) goto CRV10
CRV9$ continue
CRV10$ temp1(k) = l
expand "  $1name_designs(x,0)*"
CRV8$ continue
set_decimal_places 2
expand "//Unweighted/Relative/Values//"
through CRV11, for i = 1, until m1
y = e1(i)
expand "$1name_evaluators(y,0)*  "
through CRV12, for j = 1, until num_designs
k = temp1(j)
expand "$1Case_rv(i,k)*"
CRV12$ continue
expand "//"
CRV11$ continue
expand "Weighted/Total:      "
through CRV13, for k = 1, until num_designs
l = temp1(k)
expand "$1total_crv(l)*"
CRV13$ continue
expand "//"
comment:
f = 0.
g = 1.
call ordering(total_crv,n1,rank,f,g)
comment:
expand "Ranking:    "
through CRV14, for i = 1, until num_designs
x = temp1(i)
expand "$3rank(1,x)*"
CRV14$ continue
expand "//"
read_console
```

Linear_Scoring

```
set_decimal_places 4
dfa score 2: (1,15),(1,15)
dfa total_score 1: (1,15)
dfa weight 1: (2,25)
rs weight_table
read: $1weight(2)...weight(num_evaluators)
rs off
expand "//Filename weight_table read//"
ex transform_array
expand "Filename transform_array read//"
comment:
through LSD, for i = 1. until m1
x = e1(i)
through LS4, for j = 1. until n1
y = d1(j)
score(i,j) = weight(x)*transform(x,y)
total_score(j) = total_score(j) + score(i,j)
LS4$ continue
LSD$ continue
comment:
dfa temp1 1: (1,5)
say For display, choose 2 to 5 designs from the following list:
expand "/"
through LS5, for k = 1. until n1
y = d1(k)
expand "$3y:  $1name designs(y,0)/"
LS5$ continue
expand "//"
comment:
say Type in the number of designs to be displayed:
expand "/"
read_set console
read: $1num_designs=
expand "/"
say Type in the names of the designs to be displayed:
expand "/"
through LS6, for l = 1. until num_designs
read:  $2temp(l,0)
LS6$ continue
expand "//"
comment:
set_field_width 12
expand "Designs:         "
through LS7, for i = 1. until num_designs
through LS8, for j = 1. until n1
x = d1(j)
if ceql_F(temp(i,0),name_designs(x,0)) goto LS9
LS8$ continue
LS9$ temp1(i) = j
expand "  $1name_designs(x,0)*"
LS7$ continue
expand "//Weighted/Values//"
```

```
set_decimal_places 2
through LS10, for k = 1, until m1
y = e1(k)
expand "$1name_evaluators(y,0)*  "
through LS11, for l = 1, until num_designs
i = temp1(l)
expand "$1score(k,i)*"
LS11$ continue
expand "//"
LS10$ continue
expand "Total        "
through LS12, for j = 1, until num_designs
k = temp1(j)
expand "$1total_score(k)*"
LS12$ continue
expand "//"
comment:
f = 0.
g = 1.
call ordering(total_score(1),n1,rank,f,g)
comment:
expand "Ranking:    "
through LS13, for k = 1, until num_designs
x = temp1(k)
expand "$3rank(1,x)*"
LS13$ continue
expand "//"
read_console
```

Quasi_Separable

```
set_decimal_places 4
set_field_width 12
dfa a 3:(1,3),(1,6),(1,6)
comment: 1st dimension: level; 2nd: group designation; 3rd: values.
dfa u 3:(0,3),(1,15),(1,8)
comment: 1st dimension: level; 2nd: level evaluators; 3rd: level designs.
ex grouping_evaluators
expand "/Filename grouping_evaluators read//"
ex transform_array
expand "Filename transform_array read//"
through QS1, for i = 1. until m1
x = e1(i)
through QS2, for j = 1. until n1
y = d1(j)
u(level,i,j) = transform(x,y)
QS2$ continue
QS1$ continue
rs corner_utilities
through QS4, for i = 1. until 3.
through QS5, for j = 1. until 6.
read: $1a(i,j,1)...a(i,j,6)
QS5$ continue
QS4$ continue
rs off
expand "Filename corner_utilities read//"
through QS11, for aa = 1. until n1
i = level
k = aa
QS6$ j = 0.
mm = 0.
through QS, for ij = 1. until 6.
mm = mm + group(i,ij)
QS$ continue
ii = 1.
QS7$ j = j+1.
if (group(i,j).eql.2.) goto QS9
if (group(i,j).eql.3.) goto QS10
u((i-1),j,k) = u(i,ii,k)
goto QS8
QS9$ u((i-1),j,k) = a(i,j,1)*u(i,ii,k)+a(i,j,2)*&
u(i,(ii+1),k) + (1.-a(i,j,1)-a(i,j,2))*u(i,ii,k)*&
u(i,(ii+1),k)
goto QS8
QS10$ k1 = a(i,j,4)-a(i,j,1)-a(i,j,2)
k2 = a(i,j,5)-a(i,j,1)-a(i,j,3)
k3 = a(i,j,6)-a(i,j,2)-a(i,j,3)
k4 = 1-a(i,j,6)-a(i,j,5)-a(i,j,4)+a(i,j,1)+a(i,j,2)+a(i,j,3)
u((i-1),j,k) = a(i,j,1)*u(i,ii,k) + a(i,j,2)*&
u(i,(ii+1),k) + a(i,j,3)*u(i,(ii+2),k) + &
k1*u(i,ii,k)*u(i,(ii+1),k) + k2*u(i,ii,k)*&
u(i,(ii+2),k) + k3*u(i,(ii+1),k)*u(i,(ii+2),k) + &
k4*u(i,ii,k)*u(i,(ii+1),k)*u(i,(ii+2),k)
```

```
QS8$ ii = ii + group(i,j) + 1.
if (ii.les.mm) goto QS7
i = i-1.
if(i.grt.0.) goto QS6
QS11$ continue
comment:
dfa temp1 1: (1,5)
say For display, choose 2 to 5 designs from the following list:
expand "/"
through QS12, for k = 1, until n1
y = d1(k)
expand "$3y:  $1name designs(y,0)/"
QS12$ continue
expand "/"
comment:
say Type in the number of designs to be displayed:
expand "/"
read_set console
read: $1num_designs=
expand "/"
say Type in the names of the designs to be displayed:
expand "/"
through QS13, for l = 1, until num_designs
read: $2temp(l,0)
QS13$ continue
expand "/"
comment:
expand "Designs:          "
through QS14, for i = 1, until num_designs
through QS15, for j = 1, until n1
y = d1(j)
if ceql_F(temp(i,0),name_designs(y,0)) goto QS16
QS15$ continue
QS16$ temp1(i) = j
expand "  $1name_designs(y,0)*"
QS14$ continue
expand "//Utility/Values//"
through QS17, for k = 1, until m1
x = e1(k)
expand "$1name_evaluators(x,0)*  "
set_decimal_places 2
through QS18, for l = 1, until num_designs
i = temp1(l)
expand "$1u(level,k,i)*"
QS18$ continue
expand "//"
QS17$ continue
expand "Aggregated/Utility     "
set_decimal_places 4
through QS19, for j = 1, until num_designs
k = temp1(j)
expand "$1u(0,1,j)*"
QS19$ continue
expand "//"
dfa temp_u 1: (1,20)
through QS20, for i = 1, until n1
temp_u(i) = u(0,1,i)
QS20$ continue
comment:
```

```
f = 0.
g = 1.
call ordering(temp_u,n1,rank,f,g)
comment:
expand "Ranking:     "
through QS21, for k = 1, until num_designs
x = temp1(k)
expand "$3rank(1,x)*"
QS21$ continue
expand "//"
read_console
```

Dynamic_Control_1

```
max1 = 0.
dfa potential_designs 2: (1,30),(1,4)
dff maxlist(,20,0,1)
expand "/Type in the number of evaluators//"
read_set console
read: $1num_evaluators=
rs off
a = 1.
current_designs = 1.
ex names
expand "/Filename names read//"
ex structure
expand "Filename structure read//"
dff new_designs(,,current_designs,a)
say Determination of structure of potential designs
comment:
call new_designs(structure_designs,potential_designs,current_designs,a)
comment:
set_decimal_places 4
set_field_width 3
expand "/Potential Designs://"
a = a - 1.
through C3, for k = 1. until a
if (potential_designs(k,1).eql.0.) goto c3
expand "No. $3k:   $3potential_designs(k,1)...potential_designs(k,3)//"
C3$ continue
dfa current_value 3: (1,20),(1,5),(1,2)
comment: 1st dimension: designs, 2nd: steps, 3rd: p,theta
rs current_value_array
through C, for i = 1. until current_designs
through C0, for j = 1. until 5.
read: $1current_value(i,j,1)...current_value(i,j,2)
C0$ continue
C$ continue
rs off
expand "Filename current_value_array read//"
say Selection of best design experiment follows
comment:
ex selection
comment:
say Type in a name for the selected design (up to 10 characters
say in length):
read_set console
expand "/"
read: $2name_designs(current_designs,0)
rs off
expand "//"
store session
read_console
```

Dynamic_Control

```
dfa rank 2: (1,25),(1,20)
dfa temp 2: (1,5),(0,2)
dfa d1 1: (1,15)
dfa e1 1: (1,15)
dff maxlist (,30,0,1)
ex session
expand "/Previous session retrieved//"
dfa value 2:(2,25),(1,24)
rs value_array
through DC3, for k = 2. until num_evaluators
read: $1value(k,1)...value(k,24)
DC3$ continue
rs off
expand "Filename value_array read//"
dfa impact 2: (2,25).(1,20)
rs impact_table
through DC4, for l = 2. until num_evaluators
read: $1impact(l,2)
DC4$ continue
rs off
expand "Filename impact_table read//"
dfa u 3: (0,3),(1,15),(1,8)
comment: 1st dimension: level, 2nd: evaluators, 3rd: designs.
p = 0.
through DC5, for i = 1. until num_evaluators
if ((structure_goals(i,level).eql.0.).or.(structure goals(i,(level+1))&
.neq.0.)) goto DC5
p = p + 1.
x = impact(i,current designs)
u(level,p,current_designs) = value(i,x)
DC5$ continue
say Utility calculations for chosen design are complete
expand "/Quasi-additive utility aggregation//"
ex grouping_evaluators
dfa ab 3:(1,3),(1,6),(1,6)
comment: 1st dimension: level; 2nd: group designation; 3rd: values.
rs corner_utilities
through DC6, for i = 1. until 3.
through DC7, for j = 1. until 6.
read: $1ab(i,j,1)...ab(i,j,6)
DC7$ continue
DC6$ continue
rs off
expand "Filename corner_utilities read//"
k = current_designs
i = level
DC8$ j = 0.
mm = 0.
through DC, for ij = 1. until 6.
mm = mm + group(i,ij)
DC$ continue
ii = 1.
```

```
DC9$ j = j+1.
if (group(i,j).eql.2.) goto DC11
if (group(i,j).eql.3.) goto DC12
u((i-1),j,k) = u(i,ii,k)
goto DC10
DC11$ u((i-1.),j,k) = ab(i,j,1)*u(i,ii,k)+ab(i,j,2)*&
u(i,(ii+1),k) + (1.-ab(i,j,1)-ab(i,j,2))*u(i,ii,k)*&
u(i,(ii+1),k)
goto DC10
DC12$ k1 = ab(i,j,4)-ab(i,j,1)-ab(i,j,2)
k2 = ab(i,j,5)-ab(i,j,1)-ab(i,j,3)
k3 = ab(i,j,6)-ab(i,j,2)-ab(i,j,3)
k4 = 1-ab(i,j,6)-ab(i,j,5)-ab(i,j,4)+ab(i,j,1)+ab(i,j,2)+ab(i,j,3)
u((i-1),j,k) = ab(i,j,1)*u(i,ii,k) + ab(i,j,2)*&
u(i,(ii+1),k) + ab(i,j,3)*u(i,(ii+2),k) + &
k1*u(i,ii,k)*u(i,(ii+1),k) + k2*u(i,ii,k)*&
u(i,(ii+2),k) + k3*u(i,(ii+1),k)*u(i,(ii+2),k) + &
k4*u(i,ii,k)*u(i,(ii+1),k)*u(i,(ii+2),k)
DC10$ ii = ii + group(i,j) + 1.
if (ii.les.mm) goto DC9
i = i-1.
if(i.grt.0.) goto DC8
say Utility aggregation for the selected design is complete
show u(0,1,current_designs)
expand "/"
read_console_return
say Bayes_Posterior is called, to revise priors over the generated
say design and its parents.
ex Bayes_Posterior
dfa expected value 1: (1,20)
dfa index1 1: (1,20)
m1 = 0.
comment: m1 is the number of level 3, elemental designs
through DC15, for i = 1, until current_designs
if (structure_designs(i,3).eql.0.) goto DC15
m1 = m1 + 1.
through DC16, for j = 1, until 5.
expected_value(m1) = expected_value(m1) + current_value(i,j,1)*&
current_value(i,j,2)
DC16$ continue
index1(m1) = i
DC15$ continue
if (m1.eql.0.) goto DC20
index = 1.
comment:
call maxlist(expected_value(1),m1,max1,index)
i1 = index1(index)
set_field_width 3
expand "Current best elemental design is "
expand "$3structure_designs(i1,1)...structure_designs(i1,3)/"
expand "Expected value = $53$1max1//"
goto DC17
DC20$ expand "/No level 3 design has been generated as yet//"
DC17$ say Determination of potential new designs and selection of the
say best design experiment from among these.
expand "//"
a = 1.
dfa potential_designs 2: (1,30),(1,4)
dff new_designs(,,current_designs,a)
```

```
say Determination of the structures of potential designs
comment:
call new_designs(structure_designs,potential_designs,current_designs,a)
comment:
set_decimal_places 0
set_field_width 2
expand "/Potential Designs://"
a = a-1.
through DC18, for k = 1, until a
if (potential_designs(k,1).eql.0.) goto DC18
expand "No. $3k:  $3potential_designs(k,1)...potential_designs(k,3)//"
DC18$ continue
read_console_return
say Selection of the best design experiment follows
comment:
ex selection
comment:
c = current_designs - 1.
say Type in a name for the selected design (up to 10 characters in
say length):
expand "/"
read_set console
expand "/"
read: $2name_designs(current_designs,0)
expand "//"
comment:
store session
read_console
```

selection

```
dfa temp_structure 2: (1,20),(1,4)
comment: structure of parent design
dfa kk 1: (1,30)
comment: index of potential_design
dfa prior 1: (1,30)
comment: prior is the expected value of the parent design's current
comment: value
m = 0.
comment: derive structure for immediate parent of each potential
comment: design
through S1, for i = 1. until a
if (potential_designs(i,1).eql.0.) goto S1
m = m + 1.
through S2, for j = 1. until 4.
if (potential_designs(i,j).neq.0.) goto S3
temp_structure(m,(j-1)) = 0.
goto S2
S3$ temp_structure(m,j) = potential_desians(i,j)
S2$ continue
kk(m) = i
m1 = m
S1$ continue
expand "/"
say Structure of immediate parents for each potential desian
say determined
comment: determine immediate parent for each potential design and
comment: compute its expected current_value as the prior of the pot-
comment: ential desian
through S4, for k = 1. until m1
through S5, for i = 1. until current_designs
zap = 0.
through S6, for j = 1. until 3.
if (temp_structure(k,j).eql.structure_designs(i,j)) zap = zap + 1.
S6$ continue
if (zap.neq.3.) goto S5
n1 = i
goto S8
S5$ continue
S8$ through S7, for m = 1. until 5.
if (current_value(n1,m,2).leq.max1) goto SS
prior(k) = prior(k) + current_value(n1,m,1)*current_value(n1,m,2)
goto S7
SS$ prior(k) = prior(k) + current_value(n1,m,1)*max1
S7$ continue
S4$ continue
max = 0.
index = 1.
comment: choose desian with the highest expected prior
call maxlist(prior(1),m1,max,index)
comment:
k = kk(index)
level = 0.
```

```
urrent_designs = current_designs + 1.
hrough S9, for i = 1. until 3.
tructure_designs(current_designs,i) = potential_designs(k,i)
f (potential_designs(k,i).neq.0.) level = level + 1.
9$ continue
et_field_width 2
xpand "//Best experiment is $3potential designs(k,1)...potential_designs(k,3)//"
xpand "Expected value: $54$1max//"
xpand "Level No. $3level//"
et_decimal_places 4
eturn
```

Bayes_Posterior

```
dfa level_function 2: (1,3),(0,6)
comment: 1st dimension: levels, 2nd: steps
rs theta
through BP3, for k = 1. until 3
read: $1level_function(k,1)...level_function(k,5)
BP3$ continue
rs off
expand "/Filename theta read//"
num = 0.
n2 = n1
comment: n1 is the number of the parent of the generated design
n1 = current_designs
l = level
BP4$ num = num + 1.
denominator = 0.
through BP5, for i = 1. until 5.
xi = 0.30 + current_value(n2,i,2) - u(0,1,current_designs)
xa = xi*10. + 0.5
if ((xa.les.0.).or.(xa.geq.7.)) goto BP5
denominator = denominator + current_value(n2,i,1)*level_function(level,xa)
BP5$ continue
through BP6, for j = 1. until 5.
xi = 0.30 + current_value(n2,j,2) - u(0,1,current_designs)
xa = xi*10. + 0.5
if ((xa.les.0.).or.(xa.geq.7.0)) goto BP
current_value(n1,j,1) = current_value(n2,j,1)*level_function(level,xa)&
/denominator
BP$ current_value(n1,j,2) = current_value(n2,j,2)
BP6$ continue
l = l-1.
if (num.geq.2.) goto BP7
n1 = n2
n1 = n1
goto BP4
BP7$ if (l.les.0.) goto BP10
comment: generated design was on level 1, therefore two passes are
comment: sufficient
if (l.grt.0.) goto Bp8
comment: generated design was on level 3, therefore its level 1 parent
comment: must be determined
n1 = 1.
n2 = 1.
goto BP4
comment: generated design was on level 2, therefore the third pass will
comment: be the revision of the distribution over the universal action.
BP8$ c = current_designs - 2.
through BP9, for i = 1. until c
if (structure_designs(n1,1).neq.structure_designs(i,1)) goto BP9
n1 = i
n2 = i
goto BP4
BP9$ continue
```

```
BP10$ expand "Bayes_Posterior complete//"
return
```

ordering.pl1

```
ordering: proc(vector,upper_limit,rank,m,n);
dcl (vector(20,20),rank(20,20),upper_limit,l,m,n,t) float bin;
if (m=1) then go to label2;
/* ranking by increasing values */
           else do i = 1 to upper_limit;
                rank(n,i) = 1; end;
                do i = 1 to (upper_limit);
                t = vector(n,i); l = 0;
                    do j = (i+1) to upper_limit;
                    if t>vector(n,j) then rank(n,j)=rank(n,j)+1;
                                                 else
                    if t=vector(n,j) then do; l=l+1;
                                     if l>1 then go to label5;
                                            else
                                     do k=1 to upper_limit;
                                     if vector(n,k)<vector(n,j) then
                                     rank(n,k)=rank(n,k)-1;
                                     else; end;
                                     label5: end;
                                     else rank(n,i)=rank(n,i)+1;
                    end;
                end;
                go to label4;
/* ranking by decreasing values */
        label2: do i = 1 to upper_limit;
                rank(n,i) = 1; end;
                do i = 1 to (upper_limit);
                t = vector(n,i); l = 0;
                    do j = (i+1) to upper_limit;
                    if t<vector(n,j) then rank(n,j)=rank(n,j)+1;
                                     else
                    if t=vector(n,j) then do; l=l+1;
                                     if l>1 then go to label6;
                                            else
                                     do k=1 to upper_limit;
                                     if vector(n,k)>vector(n,j) then
                                     rank(n,k)=rank(n,k)-1;
                                     else; end;
                                     label6: end;
                                     else rank(n,i)=rank(n,i)+1;
                    end;
                end;
label4: return;
end;
```

maxlist.pl1

```
maxlist: proc(vector,upper_limit,value,index);
dcl (vector(30),upper_limit,value,index)float bin;
value = -1e30;
do i = 1 to upper_limit;
if vector(i)>value then do; value = vector(i);
                          index = i; end;
                     else;
end;
return;
end;
```

minlist.pl1

```
minlist: proc(vector,upper_limit,value,index);
dcl (vector(20),upper_limit,value) float bin;
value = 1e30;
do i = 1 to upper_limit;
   if vector(i)<value then do; value = vector(i);
                                index = i; end;
                      else;
end;
return;
end;
```

quasi_order.pl1

```
quasi_order: proc(rank,sum,reachability,num_e,num_d);
dcl (rank(20,20),reachability(20,20),sum(20,20),m,n,
    num_e,num_d) float bin;
    do i = 1 to num_e;
       do j = 1 to num_d;
          do k = 1 to num_d;
          if rank(i,j)<rank(i,k) then reachability(j,k) = 1;
                                 else;
          end;
       end;
    end;
do i = 1 to num_d;
   do j = 1 to num_d;
   sum(1,i) = sum(1,i) + reachability(i,j);
   end;
end;
return;
end;
```

# new_designs.pl1

```
new_designs: proc(structure,potential,current,a);
dcl (structure(20,4),potential(40,4) ,a,current,zap) float bin;
/* generates new designs */
if current = 1 then do; potential(a,1) = 1; a = a + 1;
                        go to P7; end;
         else;
do i = 2 to current;
   do j = 1 to 4;
   if structure(i,j)=0 then go to P3;
                       else do; potential(a,j) = structure(i,j);
                                potential(a+1,j) = structure(i,j);
                                go to P2; end;
   P3: potential(a,j) = structure(i,j) + 1;
       potential(a+1,j-1) = structure(i,j-1) + 1;
       potential(a+1,j) = 0;
       if j<3 then do; potential(a+1,j+1) = 0;
                       potential(a,j+1) = 0;
                       a = a+2;
                       go to P5; end;
              else do; a = a + 2;
                       go to P5; end;
   P2: end;
P5: end;
/* checks and eliminates designs already developed */
do j = 1 to a;
   do i = 2 to current;
   zap = 0;
      do k = 1 to 3;
      if potential(j,k)=structure(i,k) then zap=zap+1;
                                        else;
      end;
   if zap=3 then do; potential(j,1) = 0;
                     go to P6; end;
            else;
   end;
P6: end;
P7: return;
end;
```

## 7.2 Typical Output from Console Sessions

### (a) User Operations

Evaluators

Preliminary

order

display-impacts

Pareto

compare

display-transform

satisfaction

### (b) Static Evaluation

Single-Pass

Rishburn-Relative-Value

Case-Relative-Value

Linear-Scoring

Quasi-Separable

### (c) Dynamic Evaluation

Dynamic-Control-1

Dynamic-Control

```
ex Evaluators

Type in the number of evaluator names to be input

:23


Type in the name of each evaluator (maximum 10 characters) in
order, preceded by a single digit level number. For example:
0 Maxbenefit     (overall goal)
1 Financial      (1st secondary objective, under overall goal)
2 Cambridge      (1st tertiary objective, under "Financial")
3 tax_yields     (1st lower-level goal, under "Cambridge")
3 serv_costs     (2nd lower-level goal, under "Cambridge")
2 MIT          (2nd tertiary objective, under "Financial")


:0 Maxbenefit
:1 Financial
:2 Cambridge
:3 serv_costs
:3 tax_yields
:2 MIT
:3 total_cost
:3 returns
:1 Employment
:2 Numberjobs
:3 white_coll
:3 blue_coll
:3 service
:1 Socio_Envr
:2 Housing
:3 rental
:3 variety
:2 Social
:3 interact
:3 access
:2 Movement
:3 capacity
:3 parking
```

```
ex Preliminary

Type in the number of evaluators

:23


Type in the number of designs

:15


Filename impacts read

Filename names read

Programs accept single-level or hierarchically-structured
evaluation problems:
Type 1 if designs and goals both have only one level of
generality; the number of hierarchical levels otherwise.

:3

Type in the level number desired: 1, 2, or 3

:3


Filename structure read

>
```

```
 ex order

Type in the name of the evaluator to be used

:total_cost


Type in "increasing" for ranking by increasing value, or
type in "decreasing" for ranking by decreasing value.

:decreasing


Designs               Ranking


A211_3                   4

A221_3                   2

A222_3                   3

A231_3                   5

A321_3                   1



>
```

```
 ex display_impacts

For display, choose 2 to 5 designs from the following list:
               5:  A211_3
               8:  A221_3
               9:  A222_3
              13:  A231_3
              15:  A321_3

Type in the number of designs to be displayed:

:4

Type in the names of the designs

:A211_3
:A221_3
:A222_3
:A231_3



Designs:          A211_3       A221_3       A222_3       A231_3

Impacts

serv_costs     1650000.00   1300000.00   1350000.00   1900000.00

tax_yields     1500000.00   1150000.00   1300000.00   1950000.00

total_cost    84500000.00  82000000.00  83000000.00  85500000.00

returns             89.00        95.00        86.00        91.00

white_coll        1900.00      1650.00      1500.00      1900.00

blue_coll          200.00         0.00         0.00       300.00

service            150.00       100.00       125.00       200.00

rental             305.00       280.00       260.00       300.00

variety               .45          .55          .40          .50

interact              .30          .35          .40          .40

access             480.00       440.00       510.00       290.00

capacity            45.00        35.00        31.00        52.00

parking            250.00       220.00       250.00       200.00


>
```

```
 ex Pareto


Construction of Ordinal Ranking Matrix

Type in "increasing" for ranking by increasing value, or
type in "decreasing" for ranking by decreasing value, for
each evaluator:


Cambridge
:increasing


MIT
:decreasing


Numberjobs
:increasing


Housing
:decreasing


Social
:decreasing


Movement
:decreasing



Ordinal Ranking Matrix completed


Dominance check by constructing quasi-levels


Quasi-level 1: Pareto-efficient frontier

A210_2
A220_2
A230_2
A320_2


Quasi-level 2:  Dominated alternatives

A310_2


>
```

```
 ex compare

Type in the names of the two designs to be compared

:A221_3
:A222_3


                      A221_3       A222_3    Difference


Evaluators

serv_costs      1300000.00  1350000.00     -50000.00

tax_yields      1150000.00  1300000.00    -150000.00

total_cost     82000000.00 83000000.00   -1000000.00

returns              95.00       86.00          9.00

white_coll         1650.00     1500.00        150.00

blue_coll             0.00        0.00          0.00

service             100.00      125.00        -25.00

rental              280.00      260.00         20.00

variety               .55         .40           .15

interact              .35         .40          -.05

access              440.00      510.00        -70.00

capacity             35.00       31.00          4.00

parking             220.00      250.00        -30.00


>
```

```
 ex display_transform

Filename value_array read

transform_array is being appended
For display, choose 2 to 5 designs from the following list:

                 5:  A211_3
                 8:  A221_3
                 9:  A222_3
                13:  A231_3
                15:  A321_3

Type in the number of designs to be displayed

:4

Type in the names of the designs

:A211_3
:A221_3
:A222_3
:A321_3


Designs:            A211_3      A221_3      A222_3      A321_3

Transforms

serv_costs             .14         .26         .26         .40

tax_yields             .71         .54         .59         .38

total_cost             .07         .10         .10         .33

returns                .40         .53         .32         .71

white_coll             .96         .84         .80         .32

blue_coll              .20        0.00        0.00        0.00

service                .48         .34         .41         .34

rental                 .37         .44         .53         .37

variety                .56         .65         .51         .65

interact               .49         .55         .60         .64

access                 .60         .69         .60         .69

capacity               .41         .53         .59         .71

parking                .38         .44         .38         .38

>
```

```
 ex satisfaction

Type in the name of the design to be analysed
:A321_3


Design:     A321_3

Evaluator          Utility          Rank
serv_costs             .40             5
tax_yields             .38             6
total_cost             .33             9
returns                .71             1
white_coll             .32            10
blue_coll             0.00            11
service                .34             8
rental                 .37             7
variety                .65             3
interact               .64             4
access                 .69             2
capacity               .71             1
parking                .38             6


>
```

```
 ex Single_pass
Type in "increasing" for ranking by increasing value, or
type in "decreasing" for ranking by decreasing value; for
each evaluator


serv_costs
:decreasing


tax_yields
:increasing


total_cost
:decreasing


returns
:increasing


white_coll
:increasing


blue_coll
:increasing


service
:increasing


rental
:decreasing


variety
:increasing


interact
:increasing


access
:decreasing


capacity
:decreasing


parking
:decreasing
```

```
Ordinal ranking matrix completed

Dominance check by constructing quasi-levels


Quasi-level 1: Pareto-efficient frontier

A211_3
A221_3
A222_3
A231_3
A321_3


temp_impact copied into reduced_impact

READ_CONSOLE_RETURN
>store Single
```

```
 ex Fishburn_Relative_Value

Filename transform_array read


For display, choose 2 to 5 designs from the following list:

             5:  A211_3
             8:  A221_3
             9:  A222_3
            13:  A231_3
            15:  A321_3

Type in the number of designs to be displayed:

:4

Type in the names of the designs:

:A211_3
:A221_3
:A222_3
:A231_3

Designs:          A211_3      A221_3      A222_3      A231_3

Weighted
Values

serv_costs           .04         .07         .07        0.00

tax_yields           .08         .04         .05         .13

total_cost          0.00         .03         .03        0.00

returns              .02         .05        0.00         .02

white_coll           .16         .13         .12         .16

blue_coll            .05        0.00        0.00         .08

service              .04        0.00         .02         .06

rental              0.00         .02         .04        0.00

variety              .01         .04        0.00         .02

interact            0.00         .02         .03         .03

access              0.00         .02        0.00         .05

capacity             .01         .04         .06        0.00

parking             0.00         .02        0.00         .03

Weighted
Total                .42         .47         .41         .58

Ranking:               4           3           5           1

>
```

```
    Case_Relative_Value

Filename weight_table read

Filename transform_array read

For display, choose 2 to 5 designs from the following list:

             5:  A211_3
             8:  A221_3
             9:  A222_3
            13:  A231_3
            15:  A321_3

Type in the number of designs to be displayed:

:4

Type in the names of the designs to be displayed:

:A211_3
:A221_3
:A222_3
:A231_3

Designs:            A211_3      A221_3      A222_3      A231_3

Unweighted
Relative
Values

serv_costs            .13         .25         .25         .00

tax_yields            .23         .17         .19         .28

total_cost            .00         .19         .19         .00

returns               .17         .22         .14         .17

white_coll            .25         .22         .21         .25

blue_coll             .40         .00         .00         .60

service               .22         .16         .19         .27

rental                .18         .21         .25         .18

variety               .19         .22         .17         .20

interact              .17         .19         .21         .21

access                .18         .21         .18         .23

capacity              .16         .20         .23         .14

parking               .18         .21         .18         .24

Weighted
Total:                .17         .19         .18         .19

Ranking:                5           3           4           2

>
```

```
 ex Linear_Scoring

Filename weight_table read

Filename transform_array read

For display, choose 2 to 5 designs from the following list:

              5:  A211_3
              8:  A221_3
              9:  A222_3
             13:  A231_3
             15:  A321_3


Type in the number of designs to be displayed:

:4

Type in the names of the designs to be displayed:

:A211_3
:A221_3
:A222_3
:A231_3

Designs:            A211_3      A221_3      A222_3      A231_3

Weighted
Values

serv_costs             .01         .02         .02        0.00

tax_yields             .08         .06         .06         .10

total_cost            0.00         .02         .02        0.00

returns                .03         .04         .03         .03

white_coll             .08         .07         .06         .08

blue_coll              .02        0.00        0.00         .02

service                .02         .01         .02         .02

rental                 .04         .04         .05         .04

variety                .02         .03         .02         .02

interact               .03         .03         .04         .04

access                 .04         .04         .04         .05

capacity               .02         .03         .04         .02

parking                .01         .01         .01         .01

Total                  .39         .41         .40         .43

Ranking:                 5           3           4           1
```

```
    Quasi_Separable

Filename grouping_evaluators read

Filename transform_array read

Filename corner_utilities read

For display, choose 2 to 5 designs from the following list:

              5:  A211_3
              8:  A221_3
              9:  A222_3
             13:  A231_3
             15:  A321_3

Type in the number of designs to be displayed:

:4

Type in the names of the designs to be displayed:

:A211_3
:A221_3
:A222_3
:A231_3

Designs:          A211_3      A221_3      A222_3      A231_3

Utility
Values

serv_costs           .14         .26         .26        0.00

tax_yields           .71         .54         .59         .88

total_cost          0.00         .10         .10        0.00

returns              .40         .53         .32         .40

white_coll           .96         .84         .80         .96

blue_coll            .20        0.00        0.00         .30

service              .48         .34         .41         .59

rental               .37         .44         .53         .37

variety              .56         .65         .51         .60

interact             .49         .55         .60         .60

access               .60         .69         .60         .78

capacity             .41         .53         .59         .36

parking              .38         .44         .38         .50

Aggregated
Utility            .3385       .3683       .3240       .3658

Ranking:               4           2           5           3
```

```
 ex Dynamic_Control_1

Type in the number of evaluators

:23

Filename names read

Filename structure read

Determination of structure of potential designs

Potential Designs:

No.   1:     1  0  0

Filename current_value_array read

Selection of best design experiment follows

Structure of immediate parents for each potential design
determined


Best experiment is  1 0 0

Expected value: 0.439999990

Level No.:  1

Type in a name for the selected design (up to 10 characters
in length):

:A100_1


>
```

```
    Dynamic_Control

Previous session retrieved

Filename value_array read

Filename impact_table read .

Utility calculations for chosen design are complete

Quasi-additive utility aggregation

Filename corner_utilities read

Utility aggregation for the selected design is complete

u(0,1,2) = 0.32


READ_CONSOLE_RETURN
>rtd

Bayes_Posterior is called, to revise priors over the generated
design and its parents.

Filename theta read

Bayes_Posterior complete


No level 3 design has been generated as yet

Determination of potential new designs and selection of the
best design experiment from among these.


Determination of the structures of potential designs

Potential Designs:

No.  1:   1 1 0

No.  2:   2 0 0

READ_CONSOLE_RETURN
>rtd

Selection of the best design experiment follows

Structure of immediate parents for each potential design
determined
```

```
Best experiment is  1 1 0

Expected value: 0.363529406

Level No.  2

Type in a name for the selected design (up to 10 characters in
length):


:A110_2
```

## 7.3 Description of the North West Area Project

On July 10, 1969, President Howard W. Johnson announced that M.I.T. was purchasing the Cambridge property of the Simplex Wire and Cable Company, which had previously announced its transfer to Maine in 1970. (66) The Simplex property in Cambridge is 18.7 acres of land and buildings in 11 closely grouped parcels to the north of M.I.T.'s West Campus. Before its sale to M.I.T., the Simplex property paid $240,000 in yearly real estate taxes (1970) to Cambridge, and employed 600 persons, about 2-1/2% of the total manufacturing employment in the city.

M.I.T.'s purchase is located in an industrial sector of Cambridge, termed the "North West Area", hence the name of the project. This sector covers 135 gross acres (109 net acres) of industry, of which, M.I.T. properties (owned or under option, including Simplex) total 44 acres.

In his public announcement, President Johnson noted the effect that the transfer of the Simplex property to M.I.T. would have on Cambridge, particularly in terms of tax revenues and employment losses:

> "M.I.T. is acquiring the Simplex property as a resource for making further contributions to the construction of urgently needed new housing in Cambridge, and not for the

(66) H.W. Johnson; public announcement re Simplex purchase, (Cambridge, Mass., 10 July, 1969)

> expansion of M.I.T.'s academic campus. It is M.I.T.'s intention also to bring about new commercial development on the site that will add significantly to tax revenues and employment opportunities in Cambridge. All expected uses of the site will be taxable.....The site also presents an opportunity to add substantially through new commercial development to the tax revenues and to the number and variety of jobs in Cambridge....."(67)

The Simplex Advisory Committee was a 9 member group of faculty and administration set up in October, 1969, to recommend means of developing the Simplex site. It noted that the Simplex property was the only land resource available to M.I.T. with the acreage and development capacity to absorb a large quantity of the additional housing required for faculty and staff. The members recommended the development of housing for M.I.T. personnel, which would allow Cambridge to benefit from M.I.T.'s purchase, under the U.S. Housing Act of 1949, Section 112, by acquiring "credits" to apply to redevelopment projects elsewhere in Cambridge. They also expressed a preference for a mix of several small commercial activities on the site. (68)

The Corporation Joint Advisory Committee likewise stressed the development of housing for the M.I.T. community; primarily for faculty and staff, but also for visiting faculty and married

---

(67) Ibid.

(68) Simplex Advisory Committee; Considerations in the Future Development of Simplex and Related M.I.T. Properties, (Cambridge, Mass., M.I.T., Feb., 1970)

students. The members also advocated "non-polluting, labour-intensive" commercial and industrial uses for the site, and neighborhood centers both for the new development and for Cambridgeport. A "fine-grained mix of M.I.T. people and non-M.I.T. people" (69) was to be encouraged in the housing development if possible. Community involvement and integration of the project with Cambridgeport and Central Square, were also desirable. CJAC admitted though, that there were difficulties in attempting to create a residential neighborhood in the midst of an industrial sector with much noise and heavy truck traffic. The short-run conditions in the area are not conducive to residential development: the appearance of the surroundings, the presence of rail spurs, truck traffic, and the lack of access to the West Campus, (separated by railroad tracks), are all negative factors.

Further, major uncertainties in the North West Area make the planning of a comprehensive development difficult: major industries in the area may leave, (although exactly when, is uncertain); the market for commercial and office space is poor; interest rates for unsubsidized development are high; the railroad right-of-way separating the project from M.I.T. has been proposed as a possible location for the Inner Belt expressway and also for a D.O.T.

---

(69) Corporation Joint Advisory Committee on Institute-Wide Affairs; Report on Simplex and Related Development, (Cambridge, Mass., M.I.T., 5 June, 1970)

inner-city transit demonstration project; and the reactions of both the city of Cambridge (which must approve required zoning changes) and Cambridgeport residents, are uncertain.

The preparation of alternative development plans for the North West Area Project is being undertaken by the M.I.T. Planning Office, with co-ordination by a Steering Committee which also acts as a liaison with the M.I.T. Administration and community groups. The principal objectives and design alternatives as refined by the M.I.T. Planning Office, form the basis for the illustrative application of the techniques described in Section 4. Four issues are seen as crucial to the project:

(1) the assurance of adequate tax yields to the city of Cambridge;

(2) the development of a variety of job opportunities in the project;

(3) adequate housing for the M.I.T. and Cambridgeport communities;

(4) improvement of the North West Area environment.

These issues are elaborated into the hierarchical structure of goals and sub-goals illustrated in Figure No. 4.17, "GOAL STRUCTURE - M.I.T. NORTH WEST AREA PROJECT". The goal fabric is intended to be integrative of all the impacted actor groups in the project: i.e. there is no differentiation of objectives by actors.

Design variables which are presently perceived as crucial in the generation of development alternatives, fall roughly into 5 classes:

(1) Overall financing mechanisms: private developers, M.I.T., Federal and State government programs, and various combinations of these;

(2) Housing Ownership: condominium, co-operative, conventional (owning or rental) and student housing in various locations and phased combinations in the project.

(3) Programming Alternatives: the number and types of housing units, the area of commercial and office development, and community resources center;

(4) Phasing Strategies: the timing, financing and location of programmed uses, along with M.I.T. acquisition strategies for buying new properties as they become available;

(5) Locational patterns: the arrangement, mix, and density of programmed land uses, the design of open space and recreational areas, planning of parking, automobile access, and traffic flows within the project area.

Obviously, many permutations of these variables are possible; therefore, the small number of alternatives arrayed for evaluation must present as diverse and distinct a coverage of these dimensions as possible. The various grid metrics and level designations in DISCOURSE showed that some of these variables are better included at certain levels rather than others. For example, the largest grid scale (240' x 200') was most appropriate for representing contextual attributes such as community services, commercial, and Cambridgeport housing patterns; the intermediate scale was most appropriate for traffic flows, detailed population characteristics, and general land use; while the smallest grid scale (50' x 60') represented no contextual variables, but permitted housing configurations, open space design, user assignment, and ownership patterns within the project area, to be described.