Analyzing Decision Making in Software Design

Christopher W. Pidgeon
Department of Information and Computer Science
University of California, Irvine, USA
cpidgeon@luna.hac.com
pidgeon@ics.uci.edu

Technical Report 90-16
February 1990

UNIVERSITY OF CALIFORNIA
IRVINE

Analyzing Decision Making in Software Design

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Christopher W. Pidgeon

Dissertation Committee:
Professor Peter Freeman, Chair
Professor Richard Selby
Professor Allen Stubberud

1990

The dissertation of Christopher W. Pidgeon is approved,
and is acceptable in quality and form for
publication on microfilm:

_____

_____

_____
Committee Chair

University of California, Irvine

1990

Dedication

This dissertation is dedicated to my family.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost thanks are due to my advisor, Professor Peter Freeman. Without his support academically, financially, and emotionally this work would never have been completed.

I would like to thank the other members of my committee, Professors Richard Selby and Allen Stubberud, for their support and for reading my work.

To the members of REUSE and ASE research teams, Ira D. Baxter, Julio C. Leite, Guillermo Arango, Ruben Prieto-Diaz, and Y. V. Srinivas thank you for an intellectually stimulating environment in which to work. I value each friendship and look forward to a career of academic collaboration.

I am deeply grateful for the financial support received from the Division of Graduate Studies and Research of the University of California, the National Science Foundation, Hughes Aircraft Company, and the Aluminum Company of America.

Finally, thank you to my family and friends. Your patience with me is deeply appreciated.

Curriculum Vitae

1950 Born in Baltimore, Maryland

1976 B.S. Business Administration, California State Polytechnic University, Pomona.

1980 M.B.A., California State Polytechnic University, Pomona.

1983 M.S. in Information and Computer Science, University of California, Irvine

1990 Ph.D. in Information and Computer Science, University of California, Irvine
Dissertation: *Analyzing Decision Making in Software Design*

Publications

Structured Design Methods for Computer Information Systems, and Instructor's Guide with L. C. Teague. In preparation.

Structured Analysis Methods for Computer Information Systems, and Instructor's Guide with L. C. Teague. Science Research Associates, Inc. Chicago, 1985, 408pp.

"Development Concerns for a Software Design Quality Expert System," with P. Freeman. In proceedings of *22nd ACM/IEEE Design Automation Conference,* June 1985.

"Maintenance and Porting of Software by Design Recovery," with G. Arango, I. Baxter, and P. Freeman. In proceedings of *Conference on Software Maintenance-1985,* November 1985.

"TMM: Software Maintenance by Transformation," with G. Arango, I. Baxter, and P. Freeman. In *IEEE Software,* Vol. 3, No.3, May 1986.

Abstract of the Dissertation

Analyzing Decision Making in Software Design
by
Christopher W. Pidgeon
Doctor of Philosophy in Information and Computer Science
University of California, Irvine, 1990
Professor Peter Freeman, Chair

A model is given for the analysis of rationality in design decision making. We define a formal means for answering the query, *To what extent has a designer, on a particular occasion, using an explicit definition of 'good', decided rationally?*

A decision rationality classification scheme is proposed. This scheme incorporates non-compensatory decision analysis techniques (dominance and conjunctive cut-off) as well as compensatory techniques (simple and hierarchical additive weighting, linear assignment, concordance, and displaced ideal). A formal definition of design decision is derived by extending the Lehman, Stenning, Turski transformational model of the software design process. Their view of artifact specification mappings between linguistic systems is extended to include the concomitant effect of the mapping on resource expenditure.

A formal specification for decision control knowledge is defined. This representation is the union of that knowledge required to support the various decision analysis techniques. Presumed to operationalize a designer's goals, the knowledge representation scheme includes five levels:

1. Each *objective* expresses some relevant design concern for an artifact and/or resource characteristic.
2. Each *criterion* expresses some relevant decomposition of a superior objective or criterion.
3. Each *attribute* expresses the bottom-most decomposition for a superior criterion. Each attribute may have a weight indicating its relative contribution to its superior criterion.

4. For each attribute, a *value function* expresses the designer's preference ordering over observed performance for an attribute.
5. For each attribute, an *observation channel* describes an observer independent metric over some specification (either resource or artifact) rendered in some linguistic system and a procedure for application of that metric.

Our model is applied to problems in Structured Design and conceptual data modeling. We argue that a comprehensive design history must include not only the transformations applied but also the rationale used in deciding their application. This rationale must include decision control knowledge governing both artifact (product) and resource (process) facets of design decision making. The principal contribution of this work is that the opacity of the decision intensive aspects of design are reduced thereby taking a necessary step for increasing the efficiency and effectiveness of software development.

Chapter 1
The Problem and the Approach

1.1 Background

This dissertation advances a formal model for the assessment of rationality in routine software design decision-making. It represents an effort to formalize an operational technique called design rationalization [16, 54].

The objective of design rationalization (DR) is to make software designs more reviewable and thereby, to have a better chance of meeting the expectations of developers, maintainers, and users. The central question addressed by DR is,

> To what degree has a designer, on a particular occasion, using an explicit definition of 'good', decided rationally?

Design rationalization stems from the plausible, but unproven hypothesis that there is a direct correspondence between the quality of a design and the degree of rationality in the process leading to said design. DR holds that this decision-intensive process should be based on logical reasoning, supported by facts, and explicitly recorded. DR distinguishes between two kinds of designing—*discovery* and *routine*. The former is distinguished by creativity, the latter by rational choice. As presented in [16], DR focuses on routine design situations.

The cornerstone of DR is the explicit recording of information used in the course of design decision making. DR identifies two types of rationalization. One type, synthesis rationalization, is the pre- or co-structuring of information that is collected and recorded before or during the decision process. The other type, called analysis rationalization, is the post-structuring of information produced after committing to a design decision. Parnas ascribes the provocative rubric "faking" to indicate the idealization of this post-structuring [54]. Others consider keeping an incomplete design history [69], still others consider informal rationales for "significant" decisions [76].

Though acknowledged as a worthwhile thing to do, DR has seen only limited application. This is probably due to its largely intuitive basis and a lack of empirical justification for the hypothesis that more rational decision making leads to better designing and therefore, to better designs. Each of these in turn stem from the absence of a model for the central activity in designing, decision making. Lack of support for capturing and recording decisions is also a major impediment.

The discipline of decision analysis is concerned with the systematic transformation of opaque decision problems into ones which are transparent. This is achieved through the application of formal analytical techniques. An opaque decision problem is difficult to understand, solve, or explain. It is not simple, clear, or lucid. In contrast, a transparent decision problem is readily understood, clear, and obvious. Other engineering disciplines have successfully applied decision theoretic techniques in clarifying decision problems [20, 21, 75].

The research reported here represents a novel application of the decision theoretic paradigm to the software process in general and to design decision-making specifically.

1.2 Research Goal

A designer's goal-seeking characteristics in software development can be characterized as follows. Being unable to satisfactorily describe his goals in terms of *one* objective, a designer customarily maintains multiple objectives. Each is relevant to some aspect of the design artifact or design process. Two general categories of objectives can be identified: those concerned with design artifact effectiveness and those concerned with design process efficiency. The former focus on properties of specifications while the latter focus on the deployment of resources during design.

Due to their multiplicity, the objectives may frequently be in conflict with each other. When they are, a multiobjective problem exists. A particularly important aspect of the designers multiobjective decision problem is temporal. Thus, at best, the designer can only "optimize" as of that time when the decision is made. This will frequently be considered non-optimal when viewed in subsequent times e.g., due to the effects of learning.

Typically, design decision problems are so complex that any attempt to discover or impose some definitive set of optimal actions is useless. Instead, designers express their objectives such that outcomes may be deemed good enough i.e., they subscribe to Simon's principle of bounded rationality [72]. The concept of bounded rationality (which might be more appropriately named, bounded optimality) should

not be confused with the concept of satisficing. Satisficing is sometimes incorrectly characterized as the pursuit of prespecified goals (or aspiration levels) with respect given criteria. More accurately, satisficing is the consequence of "an incomplete or unsuccessful attempt at optimization." Zeleny further asserts [92]:

> The idealized concept of rationality assumes maximization of a fixed or relatively stable objective, a known set of relevant alternatives and their outcomes, and a skill in computation that allows one to reach the highest attainable point with respect to the objective.

Zeleny cautions that such an ideal rationality is unattainable due to limits on human information processing, the dynamic nature of objectives, imperfect information, and constraints on resources available for search. He concludes:

> The point is that neither maximization nor optimization is incompatible with bounded rationality. Given all the constraints and limitations indicated above, one can still pursue a given maximizing objective *subject to constraints.* Unconstrained, unbounded optimization is rarely postulated in any economic theory; it is a mathematical artifact. *Bounded optimality*, i.e., optimization under all the constraints and limitations of the human mind, would be a suitable term for human decision making.

Granting all these difficulties, we conclude that designers do attempt to reason rationally in solving their decision problems. To help, they rely on a great store of past human experience codified for them in the form of methods, techniques, and practices. These principles, maxims, and heuristics are such that adhering to them is no guarantee of success, but they do afford guidance.

The goal of this research is to combine these various normative, descriptive, and practical perspectives into a single logical framework for the analysis of software design decision-making.

Concisely, the three viewpoints which influence our work are:

- Practical design decisions are frequently ineffective without: 1) normative guidance e.g., in the form of design method and 2) descriptively accurate representations of information about the design artifact and process.
- Normative guidance is vacuous without: 1) practical problems to solve (ones with which real-world practitioners can relate) and 2) descriptive devices to gather facts and predict consequences i.e., design artifact and process specifications with concomitant reasoning formalisms.

- Descriptive results of design research findings are uninteresting without: 1) significant application to software development and 2) normative incentive such as the improvement of designs and designing.

Thus when addressing the central question of DR, we strive for that delicate balance and synergism which combines the strengths of all three points of view while attempting to avoid falling prey to the excesses of any one.

1.3 Research Problem

Informally, the problem addressed in this research is the reformulation of the DR query such that it may not only be asked in some well-defined manner, but also answered with some measure of rigor. This effort entails formalizing several related components of the design decision milieu. These include:

- distinguishing routine from non-routine design decision;
- defining the predicate "rational";
- determining the "occasion" for designing;
- establishing a knowledge representation for a designer's explication of "good"; and
- allowing for the definition of specifications for design artifact and design process resources as well as transformations affecting those specifications;

We define design from two distinct perspectives: artifact and process. By design artifact we mean the collection of sentences constituting a specification for the construction of a software-based system. Design process connotes the purposeful, human-directed activity of asserting and retracting sentences constituting a specification. We consider the elicitation of a specification setting forth the requirements for a system as separate from design. Similarly, we consider the construction of a software artifact from its design specification as separate from design. We recognize that this bounding of design is a somewhat limiting but useful abstraction from reality.

The meaning and acceptability of the sentences forming the object of design, a specification, are governed by a linguistic system [45, 81]. A linguistic system includes: 1) a grammar giving the rules for well-formed sentences; 2) a system of logic specifying logical axioms and rules of inference; and 3) a collection of extra-logical symbols and axioms related to some underlying problem domain.

Using the terminology of [45, 81] (hereafter, the LST model), a routine design decision consists of an agent who either transforms sentences constituting an artifact's

current base specification or refines those sentences into sentences in some target linguistic system. The general notion of transformation or refinement is similar to the mathematical notion of a mapping. The distinguishing feature (hence, the separate terms) being whether or not the source and target specifications are expressed in the same linguistic system.

We elected to continue use of the term "routine" from DR because it connotes the deterministic and mechanizable. However, these are not the only kinds of decision confronting the designer. These other kinds of decision are deemed non-routine owing to their meta-relationship with routine and owing to their involving some degree of creativity. They include: selecting or synthesizing a current base or target linguistic system; choosing whether to transform the current base specification or to refine to the target specification; electing to modify the current base or target linguistic system; choosing to synthesize a new linguistic system.

The LST model makes no mention of resources. Practical, real-world designing is resource constrained. It seems an unrealistic omission to exclude from our model representation of the expenditure of some precious commodity which is a direct consequence of routine decision.

Regardless of the type of decision, the effect of decision may be characterized as a change of state (as in Figure 1.1. Thus, the state-space paradigm so prevalent in AI research may be brought to bear. A decision, then, is an instance of an abstract entity completely defining the transition between two design states. By decision process we mean the activity of assessing a design state coupled with the companion activity of asserting and retracting sentences which completely define the transition to a new design state. Each transition consists of:

- specifications for both the artifact and resource states (i.e., the bases for observation)
- the library mappings (i.e., the sources for alternatives), and
- the decision control knowledge used in determining a particular choice of mapping.

The persistent record of each design state transition constitutes a design history.

In keeping with the distinction between discovery and routine kinds of design and in the interests of tractability, we confine our model to the choice of mapping whose consequences 1) directly affect specifications and 2) cause some resource(s) to be expended. As mentioned above, we call such decisions, routine. While non-routine decision types are not analyzed in our model, their effects on the design state must be represented as they may engender change in the designer's decision control knowledge.

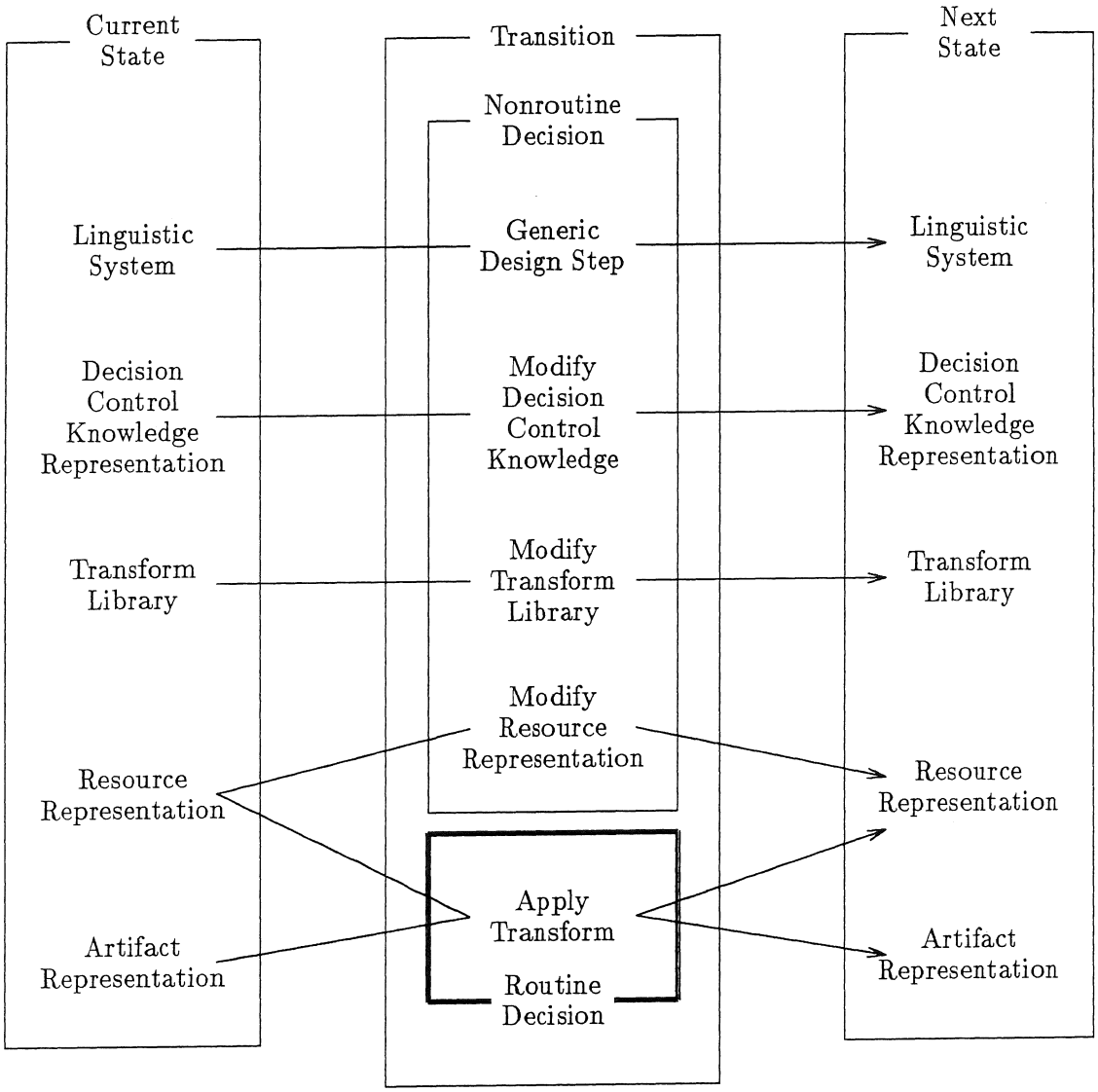

Figure 1.1: Design decision viewed as a state transition.

Decision control knowledge is a time varying structure expressing the objective and subjective controls on a designer's decision-making. It explicitly represents what the designer holds as "good". If we are to critique a designer's choices made with respect to this knowledge, it too must be represented as part of the design state. The necessity for a time backdrop stems from observations that: 1) a designer's notion of good may change and 2) decisions are serialized. Our assessment of rationality is predicated on the decision control knowledge used at the time the decision was taken.

A design rationale is the explicit representation of a designer's time-varying decision control structure coupled with the sequence of decisions (and their concomitant states) over which the decision control structure presumably had some influence.

Using the aforementioned terms, our research problem may now be stated.

> Define a model for the determination of rationality for routine design decisions predicated on the explicit representation of decision control knowledge, routine decision, and artifact and resource specifications.

We now outline our approach to the solution of this problem.

1.4 Approach

The essence of our specification for the predicate rational lies in two fundamental observations. Firstly, the range of rationality is not binary. Like [12] we presume both a lower and upper bound on rationality.

- An agent is said to be more rational when he undertakes all and only those acts that are supported by his particular decision control knowledge.
- An agent is said to be less rational when he undertakes some, but not necessarily all, of those acts that are supported by his particular decision control knowledge structure.

Secondly, though decision theorists have classified their numerous decision analysis techniques, no one has proposed the application of their classifications in a definition of rationality.

Our design decision model defines an ordinal scale of rationality. Each rationality level stems from an ordering induced by a) the classification of decision analysis techniques postulated by decision theorists and b) the observation that decision analysis techniques can be applied in series.

In Figure 1.2, noncompensatory techniques are considered weaker analyses of rationality due to the fact that tradeoffs between between decision attributes are not considered. Whereas, in compensatory techniques, tradeoffs between decision attributes are considered.

In the noncompensatory category, no distinction can be made between the dominance technique and conjunctive cutoff technique in so far as rationality assessment is concerned. However, the serial application of dominance and conjunctive cutoff is considered stronger than mere application of one or the other.

In the compensatory level, several different techniques (with at least one technique from each of scoring, compromising, and concordance category) are applied independently and in parallel. This is justified by the observation that each considers a slightly different perspective on the application of decision control knowledge in the ranking of alternatives: scoring uses multiattribute value functions; compromising techniques indicate distance from an ideal solution; and concordance techniques rank alternatives on their being in accord with preferences. The independent, parallel application of these techniques is considered weaker than the aggregation of their individual rankings into an overall consensus ranking.

Each of the aforementioned decision analysis techniques presumes more than one alternative course of action is being considered. We identify another level of rationality which is weaker than the non-compensatory assessment when there is only one alternative course of action. If that course of action results in a state which is not dominated by the present state, we call the decision trivially rational.

Before any classification of rationality can be made, a designer's decision control knowledge must be represented. That is, we must formally represent the designer's notion of "good". Figure 1.3 gives the abstract architecture for our design decision model. The particular structure given for representing decision control knowledge is strongly influenced by the requirements imposed by the decision analysis techniques supporting our scale of rationality.

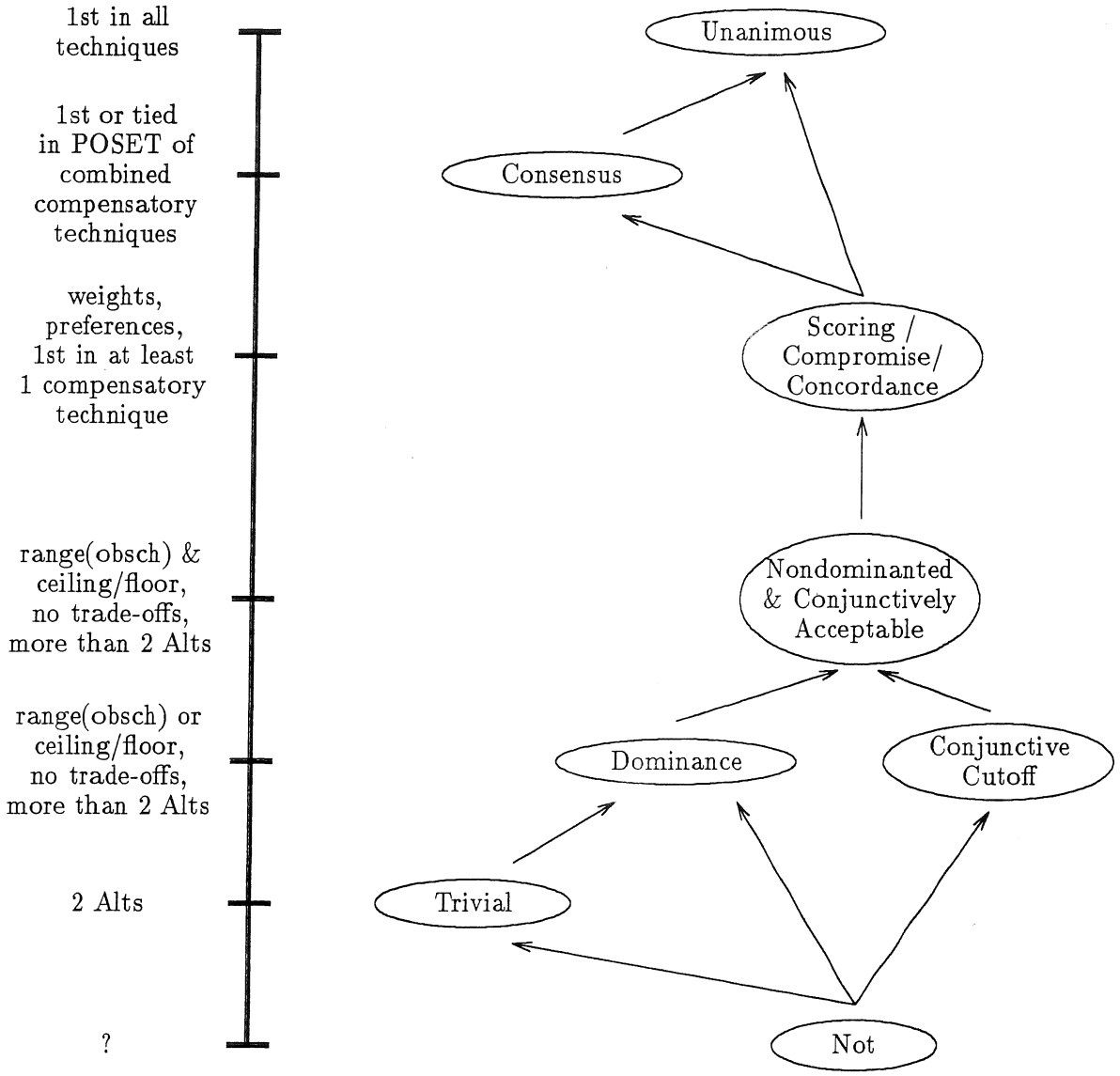

Figure 1.2: Mapping decision analysis techniques onto a scale of rationality.

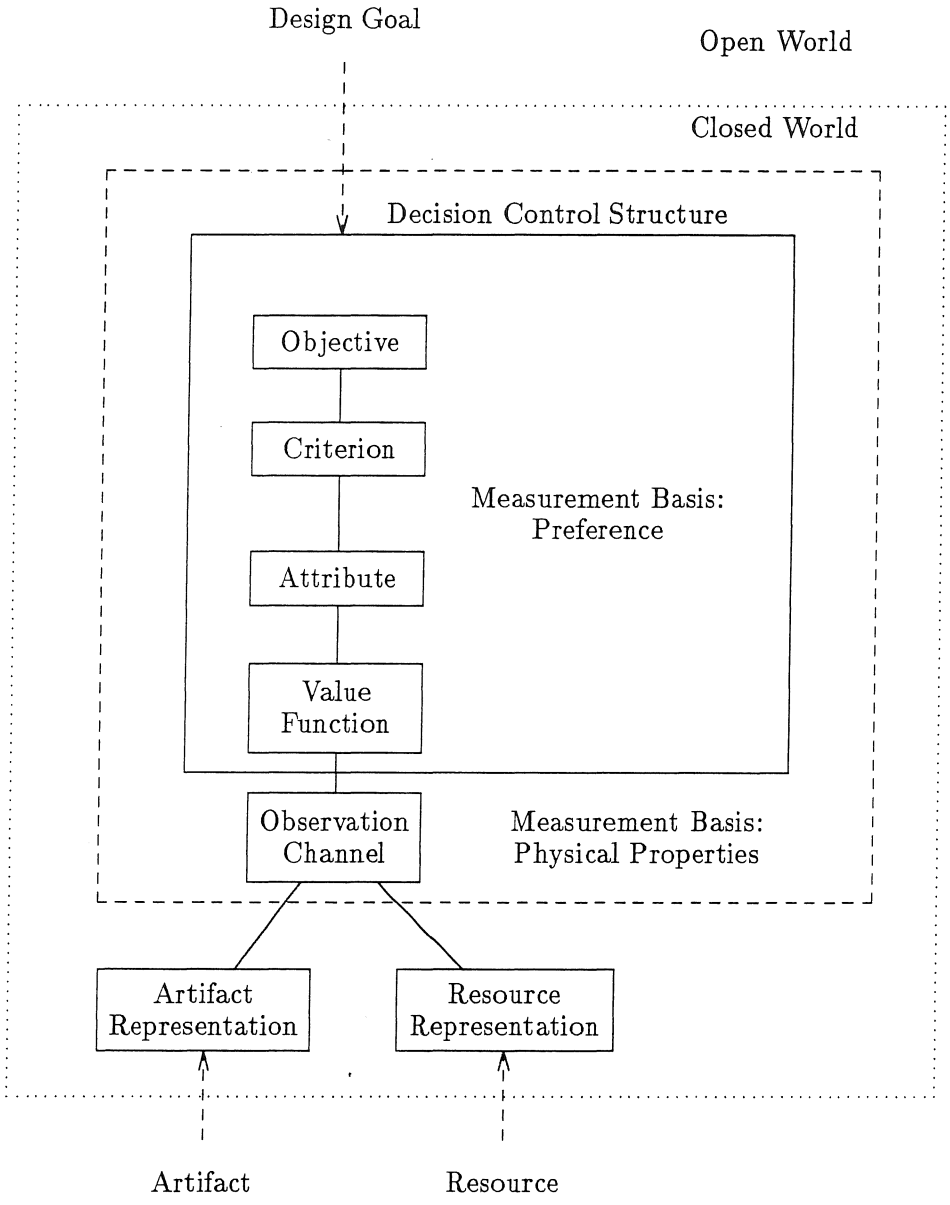

Figure 1.3: An abstract architecture for design decision modeling.

Proceeding from top to bottom, the structure consists of:

1. Each *objective* expresses some relevant design concern for either an artifact or resource characteristic. For example Structured Design exhorts us to minimize artifact complexity.
2. Each *criterion* expresses some decomposition of a superior objective or criterion. For example the Structured Design artifact complexity objective can be decomposed into module complexity and interface complexity criteria. The latter can be further decomposed into a criterion for complexity due to passed parameters and a criterion for complexity due to global data references.
3. Each *attribute* expresses the bottom-most decomposition for a superior criterion. For example, attributes of module complexity might include the McCabe's cyclomatic number, Halstead's volume metric, the total number of modules, and the total number of module invocations.
4. For each attribute, a *value function* expresses the designer's preference ordering over observed performance for the attribute. Thus the preference function for each attribute of module complexity is probably some inverse monotonically increasing function of the attribute's observation channel values. For example, as the cyclomatic number increases, preference decreases.
5. The *observation channel* for an attribute describes a metric over an artifact and/or resource and a procedure describing the metric's application. For example, a module's cyclomatic number is the number of decision nodes in the module plus one.

A crucial interface in this structure is that between the value function and observation channel. The value function serves an important purpose. It maps objective physical performance (as measured by the observation channel and therefore, with a potentially unique unit of measure) onto a scale of subjective preference (with a uniform unit of measure). Incommensurability is not an issue. In the non-compensatory techniques, direct physical performance comparison of alternatives are confined to an attribute-wise basis. For the compensatory techniques, trade-offs are only considered on the unitless (or equivalently, uniform unit) scale of preference.

Since each attribute or criterion may not contribute equally to its superior, weights may be used to indicate the relative salience of each attribute to criterion, each criterion to superior criterion, or criterion to objective.

In sum then, our design decision model incorporates the generally accepted goal for designing i.e., to synthesize a plan for construction of some artifact which must meet some specified effectiveness measures; at the same time, the effort must not waste precious design resources. This open-world goal is "operationalized" into a closed-world specification of decision control knowledge consisting of objectives, criteria,

attributes, value functions and observation channels. It is intended to represent the designer's notion of "good." With such an explicit specification we can critique the rationality of the choices and inevitable trade-offs made during the course of design. This is accomplished through the use of the ordinal scale of rationality as applied to routine design decisions.

1.5 Applying the Design Decision Model

We applied the model to one problem from Structure Design [77] and one from conceptual data modeling [6].

The application to Structured Design entailed the construction of a decision control structure approximating that of the original author. We defined a simple decision control structure consisting of a resource efficiency objective and an artifact effectiveness objective. We adopted the cost of performing a transformation as a surrogate for the amount of resource expended in transiting from one state to the next. This expenditure was measured in terms of the number of sentences added, deleted, and updated in the design specification. Clearly this is a meager substitute for resource measures which, in practice, would include personnel time and cost factors. Unfortunately such factors are generally unavailable.

The artifact effectiveness objective (Design Complexity) was decomposed into two criteria. One criterion covered intra-module complexity while the other covered inter-module complexity. This is consistent with the perspective of [89]. Module complexity was described by four attributes: the sum of the cyclomatic complexity [40] for each module, the sum of the volume metric [26] for each module, the total number of module invocations, and the number of modules. Each of these has an associated observation channel which was defined in the obvious manner.

The interface complexity criterion had two attributes—one in which inter-module communication is effected via explicitly passed parameters and the other wherein some globally shared data module is used. Substantiation for the selection of these measures of Structured Design can be found in the empirical findings of [80, 36].

In Stevens original protocol involving 13 transformations, justification is given for only one transformation at a time. That is, we are never given any indication of alternatives considered. Thus, at best, his decisions should be found to be trivially rational. In fact of the 13 decisions analyzed, only 5 were found to be trivially rational while the other 8 were deemed not rational since each had an attendant cost but yielded no improvement in the artifact complexity objective.

In order to demonstrate other levels of rationality, we synthesized several alternative transformations and considered these in a detailed manner. Our protocol is given in [56].

Finally, no resource constraints on the design process were stipulated by Stevens. He states that the design criteria and transformations can be applied in any order [77]. We suspect this will not be the case when one considers resources. A clear ordering of the transformations should emerge given our formulation of the problem as one of maximizing the "good" SD properties while minimizing the expenditure of resources in achieving those properties. In the absence of a model such as ours, it is likely that a designer is ill-prepared to deal with multiple, conflicting objectives in anything but an ad hoc manner.

In contrast to the Structured Design example, the conceptual data modeling example of [6] provided a comprehensive set of transformations, a well defined grammar for the description of extended entity-relationship models, and some insight into decision control knowledge. While potential alternative transformations were identified, the authors did not indicate which should be selected. Thus, we examined each and classified it according to our scheme.

In sum, several conclusions can be drawn from our experience with respect to presentation of designs and their derivations. Decision control knowledge is rarely articulated. Significant effort must be expended to recoup this knowledge. Encoding the knowledge for use in our model is relatively straightforward. The linguistic systems governing specifications are informal. Transformations seem to be conjured up for a particular example. These make interpretation of examples difficult and complete definition of design state transitions nearly impossible. Least we fall prey to our own castigation of published examples, the two examples completed for this work are published in their entirety in [56, 57]. Hopefully these presentations serve as examples for others and thus encourage the dissemination and use of common examples for communication between researchers in the field.

1.6 Contribution

We set out to establish a formal framework to answer the central design rationalization query,

> To what degree has a designer, on a particular occasion, using an explicit definition of 'good', decided rationally?

This effort entailed reformulating the query by giving formal definitions for each of its constituents.

The significance of our work is that the central activity in designing—deciding—which has been largely opaque is now transparent. This was accomplished by extending the LST generic design step with explicit representations for decision control knowledge and by combining this with empirically validated decision analysis techniques.

The consequences of this contribution include:

- The structure and expression of design methods can be improved through the use of our decision control knowledge formalism. This should enable both the expression and use of methodological knowledge in machines.
- The application of design methodological knowledge in a machine should leverage what is largely a manual activity. Designs can be explored more systematically by considering changes to the components of decision control knowledge, transformations, refinements, and the supporting linguistic systems.
- Program development need not be presented as merely a chain of successively refined representations with informal descriptions of the transitions between them. The value-based part of design decision can be explicitly represented and, therefore, critiqued.

We are aware of no other approach supporting such activities in the development of software.

The remaining chapters are structured as follows: Chapter 2 presents a survey of design. We begin with a general investigation of designs and designing. We focus on a particular activity, decision making, which is widely held as the essence of designing. Gaining intellectual control of this activity is a central concern of design researchers and practitioners alike. This leads quite naturally into a discussion of the role of rationality in design.

Chapter 3 focuses on decision and its analysis. Paradigms of multicriteria decision problems are explored. General and specific mathematical formulations are identified. The necessity for a time backdrop for decision modeling is identified. This stems from the fact that a decision maker's notion of "good" can change, even during the decision process. Unfortunately, nearly all the mathematical formulations assume a constant of "good". Several extant decision analysis methods are identified for use in our DDM.

Chapter 4 gives our formal presentation for determining the rationality of routine design decisions. Building from a state-transition view of the design process, we

give a definition of routine design decision. This is predicated on the assumption that some decisions (deemed non-routine) beget other decisions while others (deemed routine) only effect change in the specification of the artifact. We introduce the concept of a specification of decision control knowledge to represent that which a designer holds as "good" and uses in routine designing. It is intended to represent an operationalization of the designer's goal. It is defined in such a way that we can bring to bear the decision analysis techniques previously identified. We define each of the ordinals in our decomposition of rationality. We give an algorithm for determining where a particular decision falls on the ordinal scale of rationality.

Chapter 5 presents the application of our model to a familiar problem from Structured Design and a problem in conceptual database design.

Chapter 6 presents our conclusions by reviewing our model, contrasting it with other models of designing, and identifying future research directions.

Chapter 2
Design and Rationality

Engineering is the uniquely human activity of marshalling knowledge and artifacts from the "given-world" of the scientist along with knowledge and artifacts from the "made-world" of the engineer, combining these to make something that did not *a priori* exist, while simultaneously, and above all, obviating failure in the effort [55].

Software engineering was defined at the first major conference dedicated to the subject [50]:

> The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

This definition has been operationalized into the generally accepted goal for software engineering, 'To produce the best possible software at the lowest possible cost.' Few would deny that quality and productivity are related. *How* they are related is another matter. Some researchers argue that these are conflicting objectives [41]. Others hold that quality software necessarily costs less [35] [44].

The debate is fueled by the proliferation of practices, tools, and techniques (collectively called software technologies) each of which claims to solve some part of the software crisis. Pressman observes [60]:

> There is no single best approach to a solution for the software crisis. However, by combining comprehensive methods for all phases in software development: better tools for automating these methods; more powerful building blocks for software implementation; better techniques for software quality assurance; and an overriding philosophy for coordination, control, and management, we can achieve a discipline for software development—a discipline called *software engineering.*

A skeptical, but not pessimistic, Brooks reiterates Pressman's eschewing of a single approach—technological or managerial—which of its own accord affords "even

one order-of-magnitude improvement in productivity, in reliability, in simplicity" [9]. How then is software engineering to progress toward its goal? Brooks comments further,

> Although we see no startling breakthroughs—and indeed, I believe such to be inconsistent with the nature of software—many encouraging innovations are underway. A disciplined, consistent effort to develop, propagate, and exploit these innovations should indeed yield an order-of-magnitude improvement. There is no royal road, but there is a road.

If one accepts the premises: one, that the essential injunction of engineering is the elimination of error in designing and two, that the function of criticism is the elimination of error, then several eminent philosophers have set forth what must surely be the central concern for software engineering. The self-described pancritical rationalist, Bartley [5] asks:

> How can our lives and institutions be arranged so as to expose our positions, actions, opinions, beliefs, aims, conjectures, decisions, standards, frameworks, ways of life, policies, traditional practices, etc.—whether justifiable or not—to optimum examination, in order to counteract and eliminate as much error as possible?

Popper [58] continues,

> Nothing is exempt from criticism—not even this principle of the critical method itself.

Post concludes [59]:

> The idea is not just that there is nothing we may take on authority, or that there is nothing about which we cannot be mistaken. In addition the rational man should accept nothing which cannot be overthrown by criticism. In some sense all his beliefs must be criticizable in principle.

This chapter, indeed this dissertation, is motivated by the pursuit of pancritical rationality for software engineering.

Freeman provides a cybernetic model of software engineering in [18]. This model is intended to serve as the basis for systematic examination of software development

systems. In this effort, Freeman clearly fits Bartley's definition of a pancritical rationalist. Much of Freeman's discourse is devoted to how an organization should comprehensively examine its system development systems. Here we continue with a detailed elaboration of the design-centered issues raised by Freeman. We begin by surveying diverse viewpoints on design. We proceed to the identification of decision and its control as central acts in designing. This leads quite naturally into a discussion of the motivation for analyzing design decision making.

2.1 What is Design?

We are interested in discovering the nature of software design decision-making, especially that which distinguishes "good" designing from "bad." However, as Pye admonishes [63]: "It is not of the slightest use for us to ask 'what is good design?' until we can answer the question 'what *is* design?' "

Consulting the dictionary [84] we immediately discover, from the etymology of the word, the dichotomy of design. It stems from the Latin, "*designare,* to mark out, to define; *de*, out, from, and *signare,* to mark, from *signum,* a mark, a sign." We are at once struck by the dual nature of design: on the one hand a thing, *product,* on the other an act, *process.*

Freeman [19] presents some alternative views of the general nature of design with a collection of "one-liners" taken from [33]. He observes that the descriptions are "quite varied, but they do share the common theme of addressing the process of design, not the results." Save for one phrase in the definition attributed to Asimow, each definition provides only a unidimensional view of design. Perhaps this is due to the closeness of the disciplines of the various authors. These disciplines are distinguished by their technological bent (as opposed to social, political, environmental, etc). Yet, Asimow does introduce human needs and culture into the context of design. The terms introduced in the various definitions, *scientific principles, technical information* and *imagination* together with *economy* and *efficiency* coupled with *human needs* and *culture,* begin to polarize the various concerns for our picture of design. We see the emergence of a structure for the elements of influence that affect and effect both the products and processes of design i.e., the beginnings of a systems view of design.

Mayall [39] comments on this great diversity in design suggesting that it "conceives and defines *all* (emphasis added) the means we employ to satisfy our many and increasingly intricate needs." With this statement one begins to sense the ubiquitous nature of design. Mayall reinforces this notion stating:

> For far too long we have been afflicted by the design of so-called form and the design of so-called function, the design of the arts-based world and the design of the science-based world; a subject which might appear to confirm all too clearly that we have created a bi-cultural society. Yet so far as the real subject of design is concerned, nothing could be further from the truth. ... *design is the great integrator;* a subject in its own right and certainly not, now or ever, a derivative of art and science in whatever terms these themselves may be defined.

Pye lends support to this idea, stating [63]:

> If anyone thinks it important to civilization that a common ground between art and science shall be found, then he had better look for it in front of his nose; for it is ten to one that he will see there something which has been designed.

This integrative view of design has not been universally accepted. Cross, et. al., present a cogent argument for the distinction between design and science [13]. In a similar vein, Archer suggests that design is a third culture, a discipline distinguishable from the sciences and humanities [2]. Pugh debunks the separatist view and forcefully argues that design is not only an integrative mechanism for the arts and sciences but it must also "be considered as the culture which envelopes *both*" [62]. Pugh concludes that roles of science (rationality) and art (aesthetics) in design are not so much questions of *whether,* as they are questions of *how much.*

Perhaps the situation is best summarized by Glegg [24]:

> The rational represents the disciplined thinking applied over the entire field of design from theoretical analysis to economic realities. The inventive and the artistic, the inspirational and the intuitive must be impartially scrutinized. The rational must hold the power of veto over them all.

This view is similar in many respects to the balance of power for three branches of our government – the judiciary, executive, and legislative. In this vein, Tribus comments on rational descriptions, decisions, and designs [79]:

> The logical is the watchdog of design. Vital as this is, we must beware of going to the other extreme and regarding all design as a strictly logical exercise. It is no substitute for the inventive or the artistic. Logic may decide between alternatives but cannot be relied upon to initiate them.

The disciplines of authors cited thus far in this section are other than that of software design. Rzevski incorporates both art and science in his model of design, the Evolutionary Model of Design (EDM) [67]. Rzevski remarks that the main feature of EDM is "that it provides the means for controlling the complexity of both the design process and the system which is being designed." Rzevski's EDM has its foundation in Popper's evolutionary trial-and-error problem solving scheme as well as being strongly influenced by General Systems Theory and Cybernetics. Rzevski suggests that the EDM highlights the following important features of design.

1. Design is an investigative process (research).
2. Design is a creative process (art).
3. Design is a rational (logic-based) process.
4. Design is a decision making (value-based) process.

Thus Rzevski's EDM *explicitly* acknowledges the important roles of both art and science in software design. Jones offers an interesting comment on the role of learning (Rzevski's investigative process) in design [34].

> Designing is a highly informative process (essentially one of *un*learning what we thought *was* the case, but is *no longer true* when we have changed the situation by making something new which interacts with what was there before) and that it is wise to act always on the latest available information. 'If we'd known at the start what we've learnt while designing it, we'd never have done it like this'.

This last aphorism captures the essence of augmenting the designer's logic and/or value system. This is a recurring theme for many researchers in software development. Hoare [28], Dijkstra [15], and Gries [25] decry the current craft-like approach to software development. All suggest a more scientific approach is necessary for any significant improvement in performance.

2.2 To Design is To Decide

Freeman describes the central role of design in software engineering in [17]. With respect to design and decision-making Freeman remarks that "design is an activity concerned with making major decisions" [19]. Moreover Freeman contends, "In the last analysis, decision making is what design is all about."

Patil also characterizes design as decision-making. Mostow observes [46] that Patil distinguishes between the few, key "intellectual" decisions in design, which have

global impact, and the many, less important "mechanical" decisions, which have local impact. Dijkstra has maintained for some time that these tedious decisions can and should be mechanized [15]. Dijkstra exhorts, "Mechanizing the tedium, however, *increases* the density of the task that remains!" Others also hold that decision is a central activity in designing [7, 20, 47, 4, 69, 73].

A designer may be thought to have a basic decision making cycle analogous to that of the instruction execution cycle of a von Neumann computer – fetch, decode, execute. The designer must identify which is the next decision to address (fetch), evaluate that decision's alternatives (decode) and their ramifications, and finally commit (execute) to selection of one of the alternatives. Mostow offers a more expansive view of this process. In his call for a comprehensive model of design decision-making [47], Mostow argues for explicit representation of the processes of:

1. Framing a decision to be made.
2. Generating alternatives.
3. Establishing criteria for comparing them.
4. Evaluating the alternatives according to the established criteria.
5. Choosing an acceptable alternative.
6. Retracting the decision if it proves unsatisfactory.

Formalizing these decision-related processes is seen as a crucial step in gaining control of software design.

2.3 Controlling the Design Process

Dijkstra [15] motivates the need for control of the design process stating:

> ... the potentialities of automatic programming equipment will only bear the fruits we look for, provided that we take the challenge of the programming task seriously and provided that we realize what we are called to design will get so sophisticated, that elegance is no longer a luxury, but a matter of life and death. ... It is in this light that we must appreciate ... all efforts to discover the intellectual disciplines for controlled design.
>
> Acceptance Speech, AFIPS Harry Goode Memorial Award, 1974

Freeman [16] offers one solution – the introduction of rationality into the design process – stating:

> The basis for design rationalization is the belief that designs can be improved by making them more rational. That is, design decisions should be based on logical reasoning, be supported by facts, and be recorded.

We would interject here that not only should the *factual* bases for decisions be recorded but also the *valuational* bases need be included as well. Freeman continues:

> The cornerstone of this technique is the *explicit* recording of design information, in the form of design problems, alternative solutions, and the evaluations or arguments leading to the choice of a particular alternative.

If we accept that the designed artifact (product) is important, then it follows that we should also think the process by which the artifact came about is important also, but for different reasons. For it is the case that if we find the product lacking (Brooks might say, a flaw in the conceptual integrity of the product [8]) for any of an number of reasons, we will probably need to review the process by which the flaw came to be. We can paraphrase Weinberg, who captures the notion succinctly [85], "Neither a product nor a process view can be the entire view."

2.4 The Nature of Rationality

Simon [71] states that rationality is concerned with "selection of preferred behavior alternatives in terms of some system of values whereby the consequences of behavior can be evaluated."

Ackoff [1] implicitly reflects a similar theme in the context of problem solving:

> ... choice exists only (1) when there are at least two possible courses of action available to the decision maker, (2) where there are at least two possible outcomes of unequal value to him, and (3) where the different courses of action have different effectiveness. In other words, choice exists where the action of the decision maker makes a difference in the value of the outcome.

Ackoff tacitly assumes a rational decision maker.

Simon suggests the use of adverbs to clarify what objectives and whose values shall be used in judging rationality. Thus Simon's classification for the rationality of decision is:

objectively – fact based, i.e. correct behavior is determined by maximizing given values in a given situation where values are factually determinable.

subjectively – maximize attainment relative to the actual knowledge of the subject (i.e., decision-maker).

consciously – to the degree that the adjustment of means to ends is a conscious process.

deliberately – to the degree that adjustment of means to ends is deliberately brought about by the individual.

In subsequent discussion, we shall always refer to decisions as rational in the conscious and deliberate senses. However, we will maintain the distinction between the objectively and subjectively rational decision by using fact-based and value-based respectively.

Simon distinguishes between *value* judgments and *factual* judgments in the following manner.

> The minute decisions that govern specific actions are inevitably instances of the application of broader decisions relative to purpose and method. ... Each decision involves the selection of a goal, and a behavior relevant to it; this goal may in turn be mediate to a somewhat more distant goal; and so on, until a relatively final aim is reached.

He concludes that decisions leading to the selection of final goals will be called value judgments. Decisions concerned with the implementation of such goals are deemed factual judgments. We can characterize the valuational judgments as the "ought's" and the factual judgments as the "is'es" or "must's."

Simon's focus is on decision-making within the domain of administrative organization. We may quite naturally ask, in what manner, if any, do factual and valuational judgments enter into decision-making in the design of software.

2.4.1 Valuational and Factual Concerns in Design

Asimow distinguishes the concerns of design as being either of two kinds: *factual* or *ethical* (or value-based) [3]. Factual concerns are distinguished from the valuational because the former may be compared with reality and thus the truth of a factual concern can be tested empirically. Whereas the factual concern presents a generalization of some relevant part of reality, the valuational concern presents a generalization of the values and mores of a culture and an individual. In a grammatical sense, the

factual concerns are in the indicative mood. Asimow comments on the testability of valuational concerns:

> They are in the imperative mood, like the Ten Commandments; and like the Ten Commandments they can only be tested in a pragmatic sense. If people generally like the results, then we assume that the corresponding principles fit the ethics of our society. We might individually disagree with the particular ethics; if so, we have the right to seek, or to offer, the leadership that could persuade people to change them.

Let us consider software design from the perspectives of several "cultures" and ask, "What values and mores are imbued in the designer that affect his decision making?"

Structured Design [89] (aka Composite Design [48, 49]) holds that "good" software design is the cumulative result of a large number of incremental technical decisions. Decisions about modularity are governed primarily by concerns for inter-module dependence (coupling) and intra-module relatedness (cohesion). Other heuristics (e.g. module size, scope of effect/control, fan-in/out) also have a bearing on decisions about the modularity of a system. Categories of module cohesion and coupling are presented on a scale of preference. The basic strategy for design in this culture is, "program structure is initially derived from problem structure by analyzing data flow (not discussed here) and subsequently rearranged according to design structuring criteria."

The data structuralists, [32] and [83], suggest that system modularity is best derived from the structure of input and output data. In Jackson Program Design, the input and output streams are described using a context-free grammar. Next, a general program structure is created to match the input and output structures. When a single program structure cannot be generated due to a "clash" between the input and output structures, two or more program structures that communicate via intermediate data structures are derived. The basic strategy for design in this culture is, "program structure is matched to problem structure by a process of derivation from the problem's data structure."

Higher Order Software (HOS) [27] represents yet another culture. This one is characterized by its axiomatic definition of appropriate design constructs. The work originated out of an analysis of the most common errors detected in the design of a large-scale, real-time project. Design errors were found to lie primarily within six basic categories that are related to the way modules interfaced with each other. The originators of HOS concluded that six basic axioms could be defined which cover the manner in which a module can control another module, pass data to another module,

and use data within a module. This culture can be construed as an enhancement of Structured Design wherein additional criteria for evaluating design constructs are prescribed in an axiomatic manner.

We could comment on other design "cultures," e.g. Parnas' decomposition criterion: decompose a system such that each module hides a design decision [53], object-oriented design, or stepwise refinement. However the point should be clear: when a designer declares that he is practising a particular design method he tacitly adopts a value system to be used in decision making.

2.4.2 How does one make good decisions?

Asimow provides some insight into this question. He suggests that making good decisions is man's most difficult and crucial task [3]. Asimow states that good decisions are based on the decision maker's perception, experience, and intuition as well as the cumulative knowledge of civilization. We look to philosophy to help us answer the question of how to organize the knowledge implied by, as Asimow puts it,

> the immediate evidence of one's senses, the accumulated experience of one's lifetime, the intuitive feeling for what is proper and fitting, and the recorded wisdom of civilization.

In its literal sense, philosophy is a love of wisdom. A philosophy implies a wisdom organized to form a usable intellectual structure. It is a body of knowledge, both principles and general concepts, which supports a given branch of learning. Philosophy also includes the application of this knowledge in the domain of their relevance. The interaction between knowledge and its application – reasoning – is summarized by Asimow [3].

> Although the choice and formulation of the principles that underlie a philosophy are subject to the vagaries of the individuals who construct it, the principles themselves must nonetheless be bound by the rules of logic in applying them to the situations which are subordinate to the philosophy. The principles must, therefore, form a consistent set, so that one does not contradict another. They must be capable of expansion by logical combination and extension to form a larger body of derived principles on which discipline may find a secure foundation.

Can a philosophy of software design be made so straightforward and be bound by the rules of logic? The writings of [65], [13], and [51] would lead us to be pessimistic,

though for differing reasons. Rittle establishes the argument that significant design problems are essentially "wicked." This wickedness stems from:

- the magnitude of the solution space – the number of alternative feasible solutions is effectively infinite.
- the multivariate nature of design objectives – each design solution must be appraised on a large number of ill-defined, disparate, and conflicting criteria.
- the temporal nature of objectives – relevant criteria will change throughout the life of the designed artifact.

As mentioned earlier Cross, et. al., assert that design method and scientific method are fundamentally different. With his provocative title, "Why is design logically impossible?", O'Cathain looks at attempts (ill favored in his opinion) to draw on the history and philosophy of science to make design theory more respectable.

In [61], Protzen presents a delightful fable to introduce three principles by which to justify decisions:

- the principle of indifference or chance,
- the principle of absolute truth, and
- the principle of idoneity.

Protzen dispatches the first two principles: design and indifference or chance are antithetical and we can never know truth absolutely, only relatively. (Curiously, [34] now actively seeks chance in designing rather than dismissing it.) Protzen's last principle is intriguing for it yields a glimmer of hope for the integration of fact-based and value-based elements of decision into a single system. The idoneous is "that which is proper to, and conforms with, the ends and intentions." The advice given in the fable is: "If you do not know the true, the idoneous you shall seek."

2.5 Conclusion

With the following eloquent statement, Asimow motivates the need for an evaluative element in a philosophy [3]:

> The origin of the philosophy is empirical, but its test is pragmatic. The solutions to which it leads must be good in the sense that they are useful. But good is a relative term that needs a specific definition especially tailored for each particular situation; its value needs to be measured in a way that is peculiar to each situation. Therefore, the philosophy must include an evaluative scheme

> which guides and enables the formulation of specific criteria of goodness. This evaluative element is essentially a feedback mechanism which serves to indicate how well the principles have been applied in the particular instance and to reveal shortcomings so that an improved application of the principles can be made.

The discipline of software engineering has matured to the point where some formal feedback mechanism must be considered if we are to progress beyond the level of craft decried by [28, 15]. However as indicated by this survey of design, said feedback mechanism must accommodate both factual and valuational elements. The superiority of the factual element of design lies in its ability to provide standards for the testing of designs and design decisions; but unfortunately, these standards are rigid and absolute; they rarely admit the uniqueness of each situation. On the other hand, the valuational element allows for the relativity and flexibility attributable to individuals, but it provides no genuine testable standards of conduct. In short, we require not an entirely fact-based appraisal mechanism having standards without flexibility; not an entirely value-based mechanism having flexibility without standards; rather, a combination of standards with flexibility.

Chapter 3
Decision Analysis

Philosophers study what constitutes "good". Good decision-making is a major concern of ethics. Central to the question of goodness is the issue of what accounts for that which happens. This, in turn, is related to the "truth" of what is known (or believed). Owing to the conclusion that universal truth is approachable but ultimately unknowable, an operational philosophy of decision has evolved wherein the goodness of a decision is measured by the extent to which its consequences satisfy the decision-maker's objectives [43].

Unlike the largely nonquantitative values with which philosophers have dealt, economists focus on the *quantification* of goodness and satisfaction [72]. This focus results in a particular viewpoint on behaviors of participants in the supply-and-demand relationship of the marketplace. Thus, each participant's behavior is framed in terms of the *utility*—a measure of the power to satisfy human wants—which a commodity provides a consumer and which the production of the commodity provides the producer. Each participant's objective is to *maximize* his utility subject to limits on his resources. The rationality of each participant is defined in terms of his deployment of limited resources in the pursuit of utility maximization.

The assumptions of the economists have not gone unchallenged. Indeed, social scientists have amassed impressive evidence suggesting that people are not anywhere near the rational decision engines postulated by the economists [92]. Logicians too have entered the fracas with formalizations of preference—logical systems for the definition of and reasoning about statements of goodness [82] and models of vague fact based on fuzzy systems [91].

The nineteenth century Italian economist and sociologist, Vilfredo Pareto put forth an idea that has found widespread use in decision problems. Pareto was concerned with the problem of what principles should govern the actions of society if it is assumed that the utilities of the individuals comprising the society cannot be compared. (Utility here means the subjective value ascribed by individuals to the various goods and services available.) Under these circumstances society cannot act to achieve the greatest total utility because each individual's utility is unlike that of

any other. Pareto suggested that society should try to achieve a condition such that each individual has the maximum utility possible without subtracting anything from anyone else's utility. This condition has been labeled Pareto optimality.

The Pareto optimality problem has two distinguishing characteristics: 1) there is no common standard or measure of values between individuals and 2) there exists a multiplicity of objectives (at least one per individual) to be simultaneously satisfied. Like many other researchers, we note the similarity between the Pareto problem and our problem in design when faced with multiple, potentially conflicting design objectives, each of which is decomposed ultimately into incommensurate performance measures. In almost all decision making the multiplicity and incommensurability of criteria for judging alternatives is pervasive. This is the domain of the multicriteria decision problem (MCP).

In the remainder of this chapter we establish a terminological base for MCP. This is followed by alternative formalizations of MCP each of which stresses a slightly different viewpoint of MCP. We identify time as an important backdrop for MCP.

3.1 Multicriteria Decision Problems

In somewhat simplified terms, traditional research on decision has taken one of two distinct and quite often mutually exclusive paths [42], [72]:

- the descriptive - whose objective is to discover and describe how decisions are actually made, but without disturbing or improving the process, and
- the normative - which seeks to prescribe how decisions ought to be made (sometimes without checking real decision-makers to determine whether they are willing or even able to follow the prescriptions).

Research in software engineering can be classified similarly. The role of method is to influence the behavior of the software practitioner. The central question is: "Can systems be put forth that will *induce* software developers to adopt and follow methods?" It is irrelevant whether practitioners behave this way normally provided that ways to change their behavior can be found and they are willing and able to change. The model presented in this dissertation provides a feedback mechanism for the classification of design decisions. This is deemed essential for the influence of designers behavior. Without it we merely have the designer's claim to have followed the method. Since design methods address a multitude of design concerns it seems appropriate to look to decision theory for paradigms for multicriteria decision problems.

Multicriteria decision paradigms use a common ontology based largely on four fundamental terms. Though there are no universally accepted definitions we shall use the following (after [31]).

The term *attribute* has numerous synonyms e.g., performance parameter, component, factor, characteristic, property. The distinguishing feature common to these is the notion of means for measuring levels of achievement. Hence in this dissertation, we associate an observation channel (for the assessment of performance) and a value (preference) function (for the assessment of worth of the observed performance) with each atomic attribute. Whereas an observation channel may be empirically validated and may be used in multiple settings, a value function is more situation and/or agent specific. Moreover, two agents may agree on a means for observing an attribute of some phenomena but disagree on how that observation is valued.

An *objective* indicates something to be pursued and thus the direction of desired change. It is usually expressed with a maximization or minimization function. Objectives are intended to reflect the desires of the decision maker.

A *goal* (aka target) establishes an a priori level of aspiration which is to be met, surpassed, or not exceeded. *Constraint* is a common synonym for a goal which is established as a limit not to be exceeded. Goals are usually expressed in terms of a desired specific state in the space that represents an artifact.

The distinction between goal and objective is somewhat inconsistent in the literature. Quite often they are used interchangeably. We shall not do so here. Rather, as graphically illustrated in Figure 1.3, we reserve the term, goal, for the open-world and somewhat ethereal notion of something pursued. In contrast, we use the term, objective (as a noun), for the closed-world, specific characterization of a goal.

The basis for valuation is a *criterion* which defines an explicit measure of effectiveness. The plural, criteria, is often used to denote multiple attributes, multiple objectives, or both. We restrict a criterion to be a composition of other sub-criteria and/or attributes. Moreover, each objective is decomposed into criteria. Thus we assume a definitive hierarchy from objectives through criteria to attributes.

3.1.1 Structured Design in MCP terms

Consider the Structured Design [89] prescription for minimizing system design complexity as an objective. Other objectives include maximizing return on development resources, maximizing maintainability, and minimizing construction cost.

Inter-module coupling is one criterion providing a decomposition of the system complexity objective. Other criteria include cohesion, design shape, etc. Yourdon and Constantine assert that coupling is described in terms of four attributes. Moreover they explicitly acknowledge the different salience of each attribute stating, "In order of estimated magnitude of their effect on coupling, these are"

1. *Type of connection between modules.* The set of connection types include: `minimal, normal, pathological`. The observation procedure for this attribute is given in [89]:

   > If all connections of a system are restricted to fully parameterized (with respect to inputs and outputs) conditioned transfers of control to the single, unique activation/entry/origin/identity interface of any module, then the system is termed *minimally connected.* ... We shall call a system *normally connected* if it is minimally connected, except for one or more instances of the following:
   >
   > - There is more than one entry point to a single module, provided that each such entry is minimal with respect to data transfers.
   > - Control returns to other than the next sequential statement in the activating module, provided that alternate returns are defined by the *activating* module as part of its activation process.
   > - Control is transferred to a normal entry point by something other than a conditioned transfer of control.

   The preference function for this attribute states that, "All other things being equal, then, coupling is minimized in a minimally connected system; it is likely to be slightly higher with a normally connected system and much higher with pathological connections." Note that the modifiers suggest an nonlinear preference function.

2. *Complexity of the interface.* This is a function of the number of items present at the interface with smaller preferred to larger. This preference function appears to be a linear mapping of observed values.

3. *Type of information flow along the connection.* Here a preference ordering is given asserting: "Data-coupled systems have lower coupling than control-coupled systems, which have lower coupling than hybrid-coupled systems."

4. *Binding time of the connection.* For this attribute connections which are bound at execution time are preferred to those fixed at load time, which are preferred to binding taking place a linkage-edit time, which in turn is preferred to compilation time and finally least desirable are those fixed at coding time.

Thus we have successfully mapped the Structured Design concepts for coupling onto the terms from MCP.

Yourdon and Constantine define another orthogonal coupling objective which they call *common-environment coupling* or simply, *common coupling*. We mention this objective because other authors, notably [52, 77, 49, 60], insist on combining it with the aforementioned. Yourdon and Constantine caution that common coupling is a second-order effect and it "does not fit easily into the schema of (first-order) coupling strengths that we have already presented." Yourdon and Constantine's multiple objectives for coupling are easily supported using the MCP approach. There is no need to combine first and second-order effects. In fact, this is one of the strengths of the MCP paradigm.

3.1.2 Common characteristics of MCP

Disregarding their descriptive or normative orientations and disregarding the tremendous diversity in problem domains for their application, multicriteria decision paradigms share the following characteristics.

Multiplicity of objectives, criteria, and attributes: Each problem is formulated as a set of objectives, which are made up of criteria, each of which, in turn, may have multiple attributes.

Conflict among objectives, criteria, and attributes: The simultaneous satisfaction of objectives though theoretically possible is largely unrealizable owing to the fact that an increase in one requires the decrease of one or more others.

Incommensurable units: Performance for each bottom-most attribute may be measured in different units. Thus comparison may be difficult where there is no "obvious" conversion function. Moreover, differing units of measure may have no "obvious" composition function.

Both analytic and synthetic: The problem may require the synthesis of one or more alternative solutions, the analysis of some previously specified set of alternative solutions, or some combination of both.

In the following section we briefly review the mathematical formulations for MCP. This is followed by taxonomy of decision methods.

3.2 Formal Multicriteria Problems

The general mathematical form for the MCP is

$$\text{Maximize:} \quad [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]$$

$$\text{Subject To:} \quad g_i(\vec{x}) < 0, i = 1, \ldots, m$$

by choosing appropriate values for $\vec{x}$ where $\vec{x}$ is an n dimensional variable vector. Thus, the problem consists of n decision variables, m constraints, and k objectives. Any or all of the functions may be nonlinear. This form is sometimes called the vector optimization problem [11].

There are two strategies for solving the MCP. One strategy attempts to optimize a single selected objective while considering the others as part of the constraint set. The optimal solution would then satisfy these other objectives to at least some predetermined level. Thus,

$$\text{Maximize:} \quad f_i(\vec{x})$$

$$\text{Subject To:} \quad g_j(\vec{x}) < 0, j = 1, \ldots, m$$
$$f_l(\vec{x}) > a_l, l = 1, \ldots, k \wedge l \neq i$$

where a_l is a prespecified aspiration level for objective l.

The principal difficulty with this approach is choosing a_l's which result in a nonempty solution set. Implicit in this formulation of the problem is a trade-off between f_l and f_i, i.e., when $f_l \geq a_l$, the trade-off is 0, when $f_l < a_l$ the trade-off is ∞. This probably does not reflect the actual value structure of most decision makers. Moreover, this value structure is sensitive to the level of a_l.

The other strategy for solving the MCP attempts to optimize a synthetic-objective comprised of the vector sum of the product of each objective and some "appropriate" weight.

$$\text{Maximize:} \quad \sum_{i=1}^{k} w_i \cdot f_i(\vec{x})$$

$$\text{Subject To:} \quad g_i(\vec{x}) < 0, i = 1, \ldots, m$$

The weights may be normalized by $\sum_{i=1}^{k} w_i = 1$.

The principal difficulty with this approach is the determination of the weight coefficient for each term of the synthetic objective. Each weight is likely to be sensitive to the particular objective as well as the levels for each of the other objectives.

3.2.1 Formulae and representations for MCP

Several equivalent notations can be made for the vector optimization formulation of a multicriteria decision problem [31, 30].

Firstly, there is the enumerated objective function form.

Given: $\vec{x}$, k objectives, m constraints, n decision variables, and an n-dimensional vector

$$\text{Maximize:} \quad [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]$$

$$\text{Subject To:} \quad g_j(\vec{x}) \leq 0, j = 1, \ldots, m$$

Equivalently, there is the vector form,

$$\text{Maximize:} \quad \vec{f}(\vec{x})$$

$$\text{Subject To:} \quad \vec{g}(\vec{x}) \leq \vec{0}$$

The interpretation of $\leq$ is, for any two vectors $\vec{x}$ and $\vec{y}$,

$$\vec{x} \leq \vec{y} \Leftrightarrow \forall i, x_i \leq y_i.$$

The constraints $\vec{g}(\vec{x}) \leq \vec{0}$ define a *feasible set* X of the set of decision variables which satisfy the constraint set. Thus, $X = \{\vec{x} | \vec{g}(x) \leq \vec{0}\}$.

For each element in the feasible set X, there is an associated $\vec{f}(\vec{x})$ vector which maps X into a set S of *functional values.* Thus, $S = \{\vec{f}(\vec{x}) | \vec{x} \in X\}$.

The equivalent notations, then, are:

1. Maximize: $\vec{f}(\vec{x})$ Subject To: $\vec{g}(x) \leq \vec{0}$
2. Maximize: $\vec{f}(\vec{x})$ Subject To: $\vec{x} \in X$
3. Maximize: $\vec{f}(\vec{x})$ Subject To: $\vec{f}(\vec{x}) \in S$

In conclusion, the multicriteria decision problem may be concisely represented in matrix form. In the $m \times n$ decision matrix D, each $x_{i,j}$ element represents the evaluation or value for alternative i, denoted A_i, with respect to attribute j, denoted X_j. For each of the m alternatives, a row vector $\vec{x}_i$ represents the performance of alternative i on each of the n attributes $A_i, i = 1, 2, \ldots, m$ is $\vec{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$. Each column vector $\vec{x}_j$ represents the contrasting performance of each alternative with respect to attribute j, $X_j, j = 1, 2, \ldots, n$ is $\vec{x}_j = (x_{1,j}, x_{2,j}, \ldots, x_{m,j})$. We use the matrix form for our approach to design decision representation.

3.2.2 MCP categories

In distinguishing MCP, Hwang and Yoon identify multiple attribute decision making (MADM) as largely analytical with trade-offs expressed for a limited number of predetermined alternatives [31]. In contrast, multiple objective decision making (MODM) methods are characterized as synthetic since the issue confronting the decision maker is the generation of some "best" alternative subject to constraints.

In contrasting solution types for MADM vs. MODM, we note that an *ideal* solution in MADM is characterized as *subjective*. Whereas, for MODM the *optimal* solution is said to be *objective* [31]. Thus for the general form of a MODM problem,

$$\text{Maximize} : \forall x \in X, [f_1(x), f_2(x), \ldots, f_k(x)], X = x | g_i(x) \leq 0, i = 1, 2, \ldots, m$$

the optimal solution is that which simultaneously optimizes each objective function:

$$\text{Maximize} : \forall x \in X, f_j(x), j = 1, 2, \ldots, k.$$

The optimal alternative is defined as

$$A^* = (f_1^*, f_2^*, \ldots, f_j^*, \ldots, f_k^*).$$

where each f_j^* denotes the feasible, optimal value for the j^{th} objective function. Given the assumption of conflict among objectives, the optimal solution is generally infeasible.

In contrast, the ideal solution in MADM is the hypothetical alternative whose Cartesian product is composed of the *most preferable* values for each attribute in the decision matrix. That is, the ideal alternative is

$$A^* = (x_1^*, x_2^*, \ldots, x_j^*, \ldots, x_k^*)$$

where

$$x_j^* = \forall i, \max V_j(x_{i,j}), i = 1, 2, \ldots, m$$

and V_j is the j^{th} value function. This ideal solution too does not generally exist. However, it is useful as an anchor point in the decision alternative space.

We note that our approach to design decision modeling incorporates *both* the MADM notion of ideal alternative and the MODM notion of optimal alternative. We think this is necessary for a comprehensive treatment of software design decision modeling. In addition further classification of MCP solution types are required in screening alternatives.

Nondominated solutions: The "best" solution will be found here. This set is also know as noninferior, efficient, and Pareto-optimal. Membership is determined by: 1) the solution must be feasible, and 2) if there does not exist another feasible solution which will yield an improvement in one objective/attribute without causing a degradation in at least one other objective/attribute. This concept is used as the sufficient condition for the "final" solution. Thus, it is quite often deployed as an initial screen in a multi-step procedure. Formally, $\vec{x}^*$ is a nondominated solution iff $\neg\exists\vec{x} \in X$ such that $F_i(\vec{x}^*) \leq f_i(\vec{x})\forall i \wedge f_j(\vec{x}^*) \leq f_j(\vec{x})$ for at least one j.

Satisficing solutions: As defined by [72], these solutions constitute a subset of the feasible set which exceed the aspiration levels (minima or maxima) established for each attribute. Thus, these are "acceptable" solutions and they need not be nondominated. This solution procedure is deemed to be practical given the bounded rationality generally ascribed to decision makers. This approach may be used to initially screen the acceptable from the unacceptable.

Preferred solutions: The preferred solution is that nondominated solution selected as the "final" choice from the candidate nondominated solutions. As such it reflects the application of the decision maker's preference structure.

In conclusion, the generic multicriteria decision problem incorporates four components [90]:

1. the set of alternatives, X, which are input to the decision process;
2. the set of criteria, $f = (f_1, \ldots, f_q)$, controlling the decision process;
3. the outcome of each choice—$f(x) = (f_1(x), \ldots, f_q(x))$—the totality of which is denoted, $Y = f(x)|x \in X$;
4. the decision maker's preference structure which allows the determination of y^*, the "best" outcome in Y, and thus the "obvious" choice, $y^* = f(x^*)$.

In the following section we review the taxonomy of decision methods which serves as the basis for our rationality predicate.

3.3 A Taxonomy of Decision Methods

As indicated in Table 3.1 there are two major categories of multiattribute decision methods. The distinction between the two is based on whether tradeoffs are allowed between attributes. Hence, the two categories are named, compensatory

Characteristic	Non-compensatory		Compensatory		
			Scoring	Compromising	Concordance
Decision Method	Dominance Sieve	Conjunctive Cutoff Sieve	SAW HAW	TOPSIS	ELECTRE LAM

Table 3.1: A taxonomy of decision analysis methods.

(tradeoffs permitted) and noncompensatory (tradeoffs not permitted). In this section we summarize further distinguishing characteristics of each category and describe the decision methods used in our model.

3.3.1 Noncompensatory Methods

Methods in the noncompensatory category compare alternatives on an attribute-by-attribute basis. Furthermore, an alternative with inferior performance on one attribute cannot be offset by superior performance on another. These methods are considered simple yet intuitively appealing to decision makers whose knowledge and information processing ability are bounded. The noncompensatory methods do not require the articulation of preference, hence their simplicity. Accordingly, these methods yield objective results rather than subjective ones.

Two methods we shall use from the noncompensatory category are the dominance sieve and the conjunctive constraint sieve (also known as satisficing). Since a dominated alternative cannot possibly be in the set of rational final choices and since the dominance checking procedure is rather simple, we first check whether the designer only pursues nondominated alternatives. The conjunctive sieve is also a simple method for quickly dismissing alternatives which do not meet prescribed attribute performance levels. Our second rationality check then, is to determine if the designer only pursues acceptable alternatives. For other methods in the noncompensatory category see [31].

The Dominance Sieve

An alternative is considered dominated if there exists another alternative which excels it in at least one attribute and at least equals it in the remainder. The sieve of dominance produces a set of nondominated alternatives. Note that this method does not require any transformation of attribute performance (scaling) as each attribute's comparison is done on its own scale. Moreover the dominance sieve procedure does

not require further assumptions e.g., minimum or maximum performance levels for attributes.

The dominance sieve procedure can be described informally: Compare two alternatives on an attribute-wise basis. If one is dominated by the other, discard the dominated one. Consider each of the undiscarded alternative(s) with the other alternatives (not yet considered) discarding any dominated alternatives.

The Conjunctive Sieve

Simon describes a method called "satisficing" where minimally acceptable standards of performance (prescribed by a decision maker) are used to screen alternatives [72]. The minimally acceptable attribute values are called the cutoff values. They play a crucial role in the elimination of noncontending alternatives. Suppose the cutoffs are set unrealistically high, then the set of acceptable alternatives may be empty. Conversely, if the cutoffs are too low, the set of acceptable alternatives may equal the set of candidates. In which case, resources expended on the conjunctive sieve procedure are wasted. Sometimes an iterative approach is used to arrive at "appropriate" cutoff values. Because of its strong intuitive appeal, the satisficing method has enjoyed widespread use.

We shall use the conjunctive sieve to dichotomize alternatives into acceptable/unacceptable categories. An alternative A_i is classified as acceptable only if

$$x_{i,j} \geq x_j^0, j = 1, 2, \ldots, n$$

where x_j^0 is the cutoff level for attribute x_j.

As with the dominance sieve, the conjunctive sieve does not require attribute values to be expressed in numerical form nor does it require that the relative importance of attributes be expressed. It is noncompensatory. No acceptable alternative is credited for especially high attribute performance, nor is any attribute penalized for minimally acceptable attribute performance.

3.3.2 Compensatory methods

Recall that the compensatory methods accommodate tradeoffs between attributes. Thus a (perhaps small) change in one attribute may be offset by compensating (perhaps large) change in one or more other attributes. Furthermore, the rate of exchange need not be constant i.e., it may be nonlinear. Each compensatory method ascribes a single figure of merit to each alternative. This figure of merit is

intended to be a compilation of the alternative's performance on each of the multiple attributes. The compensatory methods may be classified into three subgroups according to how the figure of merit is determined.

Scoring Methods

These methods use multiattribute value functions to determine the figure of merit. There are several techniques in this category. However, they all share three common assumptions: 1) additive value functions which are 2) monotonic, and each of these two assumptions is based on 3) preference separability (decomposability). For rigorous definitions see [90]. The additivity, monotonicity, and separability assumptions for value functions are not entirely uncontroversial. See [37, 68] for a comprehensive treatment. Each of these assumptions is imposed in direct consideration of resource limits on information processing i.e., both from the decision maker's perspective of assessing weights for many attributes and the computational complexity of combining these to arrive at a figure of merit.

A preference structure with n attributes is said to be decomposable if there exist real-valued functions $v_1, \ldots, v_n$ defined on sets of possible attribute values $X_1, \ldots, X_n$ and v defined on X, such that for any $\vec{x}'$ and $\vec{x}'' \in X$

$$\vec{x}' \mathcal{P} \vec{x}'' \iff v[v_1(x_1'), \ldots, v_n(x_n')] \geq v[v_1(x_1''), \ldots, v_n(x_n'')]$$

where $\mathcal{P}$ is read, "is preferred to."

The general forms for $v(\vec{x})$ are then [11]:

- simple additivity
$$v_1(x_1) + v_2(x_2) + \cdots + v_n(x_n)$$
- weighted additivity
$$w_1 v_1(x_1) + w_2 v_2(x_2) + \cdots + w_n v_n(x_n)$$

  where $w_i > 0$ and $\sum_{i=1}^{n} w_i = 1$

The weights are used to adjust for the relative salience of each term in the value function. A method for determining a hierarchy of criteria and their weights is given in [68]. The approach is a vertical extension of simple additive weighting (SAW). It is suggested as a means for simplifying the information overload faced by the decision-maker attempting to assign weights to many criteria. It is called hierarchical additive weighting (HAW).

Compromising Methods

The compromising methods indicate which alternatives are close to an ideal. The TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) method [88] and [31] is distinguished from others in this category in that it discriminates not only on the notion of distance from an ideal alternative but also on the distance from a negative-ideal. Thus, "best" is defined as that alternative which is closest to the positive-ideal *and* simultaneously farthest from the negative-ideal. The ideal and negative-ideal alternatives—that which is composed of all best (worst) possible attribute values—are considered artificial in that each is practically unrealizable. For example, one could think of the ideal as the infinite capacity, zero cost alternative, whereas the negative-ideal is zero capacity and infinite cost. TOPSIS also distinguishes between "benefit" criteria (those in which larger attribute outcomes are preferred) and "cost" criteria (those in which smaller attribute outcomes are preferred).

TOPSIS consists of six steps:

1. Construct a decision matrix.
2. Produce the weighted normalized decision matrix.
3. Determine the ideal and negative-ideal alternatives.
4. Calculate the n-dimensional Euclidean separation between each alternative and the ideal; between each alternative and the negative ideal.
5. Calculate the relative closeness to the ideal for each alternative.
6. Rank the alternatives based on relative closeness to the ideal.

The notion of distance is fundamental to both simple additive weighting (SAW) and TOPSIS. SAW uses the city block measure [14]. TOPSIS uses the Euclidean measure. SAW is shown to be a special case of TOPSIS [31]—TOPSIS equivalent to SAW when TOPSIS uses the city block measure. This subtle change yields a dramatic change in consequences.

Whereas the city block distance measure guarantees that any alternative which has the shortest distance to the ideal (and also has the longest distance to the negative ideal), the Euclidean distance measure makes no such guarantee. SAW implementations exploit the guarantee, which is why they are simpler. More importantly, however, SAW and TOPSIS will, in general, produce different ranking of alternatives. Thus we need to include both methods in a comprehensive review of alternatives.

Concordance Methods

The concordance methods order alternatives based on their being in accord with the decision maker's preference structure. The permutation method [31] measures the level of accord with a *complete* preference order. The ELECTRE method first determines the accord based on a *partial* order of the alternatives and then constructs a possible aggregate order. Both methods can be used with cardinal and ordinal preferences. However, the permutation method was originally designed for use with ordinal preferences, whereas ELECTRE was intended for use with cardinal preferences.

The linear assignment method (LAM) ranks alternatives based on attribute performance and relative weights. It is attractive since it uses only ordinal attribute data (thus, eliminating the need to scale attributes). Moreover, the tedious trade-off analyses of other methods is not required. However [31] argue that in its base form, the method is not truly compensatory. The final rank of an alternative is determined by summing its own attributewise ranks *without* simultaneously considering all the other alternative attributewise ranks. They make a rather simple amendment to the method to make it compensatory. The problem is then stated as a linear programming model.

The ELECTRE method (Elimination et Choice Translating Reality) [31] uses a multistage procedure to conduct a pairwise comparison of alternatives. This comparison is based on a more refined notion of dominance and uses the concept of an 'outranking relationship'. Briefly, the outranking relationship between two alternatives states that the decision maker somehow accepts that one alternative is almost surely better than the other alternative. Successive comparison of the outranking relationships among alternatives yields a partial order over the alternatives. This comparison is based on the degree to which evaluations of the alternatives and the preference weights confirm or contradict the dominance relationship. This is accomplished by examining one, the degree to which the preference weights are in agreement with (concordance) the pairwise dominance relationships and two, the degree to which weighted evaluations differ from each other (discordance).

The steps in ELECTRE are:

1. Construct a normalized decision matrix.
2. Produce the weighted normalized decision matrix.
3. For each pair of alternatives, divide the set of criteria into the concordance subset and its complement, the discordance subset. The concordance set consists of those criteria for which one alternative is preferred to the other.
4. Calculate the concordance matrix reflecting the relative preference between pairs of alternatives.

5. Calculate the discordance matrix.
6. Determine the concordance dominance matrix.
7. Determine the discordance dominance matrix.
8. Combine the concordance and discordance dominance matrices into a single aggregate dominance matrix
9. Eliminate from further consideration those alternatives which are dominated in the aggregate dominance matrix.

Some observations are in order. Firstly, note that steps one and two in ELECTRE are identical to those of TOPSIS. Secondly note that the concordance and discordance matrices are, in general, asymmetric. Thirdly, note that whereas the concordance matrix represents differences among the weights, the discordance matrix represents differences among attribute values. Final selection should be based on the *net* concordance dominance value being at a maximum and the *net* discordance dominance value at a minimum. The net concordance dominance value is defined to measure the degree to which the total dominance of a given alternative exceeds the degree to which all other alternatives dominate the given alternative. A similar definition holds for the net discordance dominance. In ELECTRE, the outranking relationship is used as a surrogate for the decision maker's value function. It is less formal and weaker in terms of ordering strength.

In conclusion, we note that each decision method category emphasizes some salient feature of decision analysis. Moreover, to only apply a single decision analysis method in the determination of a course of action is to disregard the importance of multiple viewpoints. A comprehensive model of design decision must take the multiple viewpoint approach.

3.4 A Time Backdrop for Decision Modeling

Most decision theorists impose simplifying assumptions on the components of a decision model. The most stringent is the assumption that the decision control knowledge does not vary with time.

1. decision criteria are fixed and foreknown;
2. these criteria are mathematically formulated such that a single objective function to be maximized or minimized can be identified;
3. the decision maker's perceived payoffs, measured in terms of the criteria and resulting from the decisions, are preknown and deterministic;
4. the decision maker's preferences are stable and constant.

Under these assumptions there is clearly no room for modeling the effects of learning. Moreover, such constancy is quite unrealistic for design in general and software design in particular.

Although each of the components of decision do vary with time in the real world, we are aware of only one researcher in decision analysis who has attempted to incorporate a time basis into his model. Understandably, this basis is limited to those cases where the decision control knowledge components have become stabilized thus making application of decision analysis methods feasible.

Yu suggests that each of these components of decision be subscripted by time [90]. Thus, an alternative can be said to be $[t_i, t_j]$-optimal, iff it is a member of the set of alternatives considered in time interval $[t_i, t_j]$ and the alternative is the unique, nondominated alternative with respect to decision criteria applicable in time interval $[t_i, t_j]$.

Generalizing on Yu's idea, we shall always speak of a relative rationality. That is, our analysis of rationality will always be made with respect to:

- the decision control knowledge in place at the time the decision was taken and
- the alternatives explicitly considered at that time.

Thus we have defined the notion of decision occasion. A change of occasion is said to happen whenever any part of the decision control knowledge is altered or when, by the addition or removal of a mapping, a different set of alternatives is considered.

3.5 Conclusion

This chapter has explored the terminology and a taxonomy of decision analysis methods. The argument was put forth that, owing to the unique viewpoints, assumptions and limitations offered by methods in each of the taxonomic categories, no single method can be applied and still meet the requirement for comprehensive decision analysis. Additionally, the argument was made for giving a time backdrop to the components of decision. In the next chapter we show how these concepts are integrated into our model of design decision.

Chapter 4
A Formal Model of Design Decision Making

This chapter gives a formal model for determination of the rationality of routine design decisions. It provides an architecture for merging a state-transition view of the design process with decision analysis methods in the assessment of design decision rationality. The specification of decision control knowledge serves as the central girder in this architecture and is supported by the explicit representations of the design state. A scale of rationality is defined based on the type and comprehensiveness of the decision analysis method applied in judging rationality.

4.1 A State-Transition View of Design

Our view of design is developed in a sequence of three specializations (see Figure 4.1):

1. The design process can be viewed as a state-transition system wherein states embody relevant knowledge and transitions correspond to changes in that knowledge.
2. Transitions are specialized to the Lehman, Stenning, and Turski (LST) [45, 81] generic design step and specifications are added as explicit parts of the design state.
3. Design state knowledge is further partitioned to include linguistic systems and specifications for both decision control knowledge and resources. In addition, prototype transitions are defined for routine and non-routine decisions.

Thus these specializations culminate in a formal definition of routine design decision as a particular kind of state transition i.e., one in which our closed-world assumption holds.

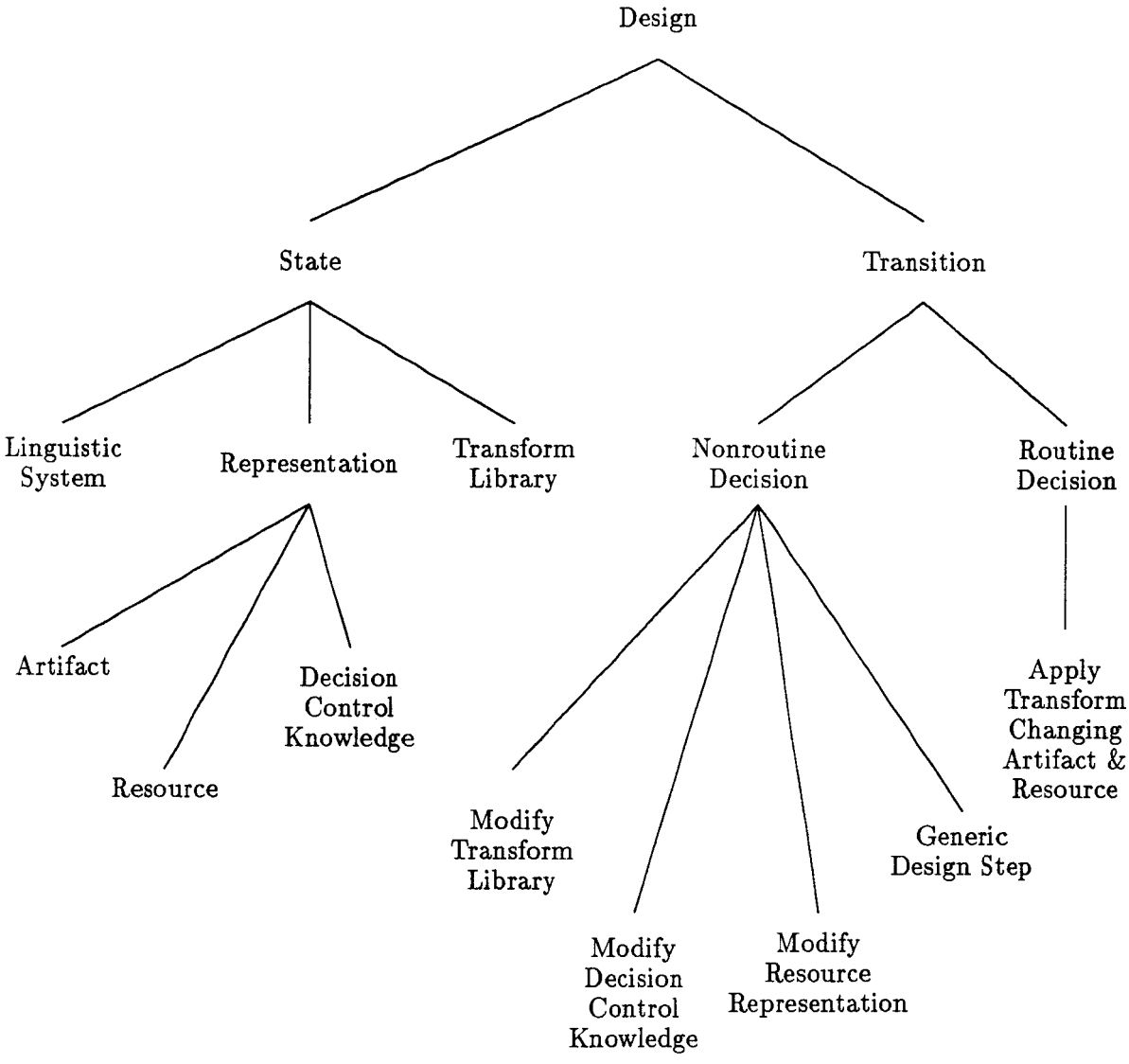

Figure 4.1: A three-level specialization of design.

Specialization 1 We begin with the assumption that there exists some kind of design process. We postulate that each step in the design process can be viewed as a state-transition system:

$$DSK \times D \longmapsto DSK' \tag{4.1}$$

where DSK is the design state represented as some structured space of knowledge and D is a decision (a transition) which induces some change in that knowledge.

Specialization 2 Here a portion of the design state and decision are distinguished as in according to the LST generic design step:

$$\langle LS, DSK_{other} \rangle \times (GDS \cup D_{other}) \longmapsto \langle LS, DSK_{other} \rangle' \cup D_{other} \tag{4.2}$$

where

LS is some linguistic system.

DSK_{other} is all other design state knowledge.

GDS is a generic design step (defined below).

D_{other} is all other kinds of design decision steps.

Thus this specialization can be characterized as a partitioning of the design state knowledge

$$DSK = LS \cup DSK_{other}$$

and a partitioning of the state transitions

$$D = GDS \cup D_{other}$$

In [45, 81] the generic design step is defined to be the construction of a map from one linguistic system to another by the composition of two actions:

1. the *extension* of a target linguistic system linguistic system e.g.

$$LS_{array,Nat} \stackrel{extend}{\longrightarrow} LS_{array,Nat,types}$$

2. the *interpretation* of a current base linguistic system in terms of the extended target linguistic system

$$LS_{stack} \stackrel{interpret}{\longrightarrow} LS_{array,Nat,types}$$

Thus, in this example the stack linguistic system is interpreted in terms of the array and natural numbers linguistic system extended by types. The generic design step consists of one, the extension of the target linguistic system of arrays and natural numbers with types and two, the current base linguistic system of stacks is interpreted in terms of this extended target linguistic system.

In the second part of this specialization step specifications are added to the state and transform steps (the means for changing a specification) are distinguished as types of state transitions.

$$\langle LS, S, DSK_{other} \rangle \times GDS \cup TS \cup D_{other} \longmapsto \langle LS, S, DSK_{other} \rangle'. \qquad (4.3)$$

The specification S is a collection of well-formed sentences in the language of some linguistic system i.e., $S \in L(LS)$. Thus a specification is given a concrete syntax and explicit semantics. The transform step TS maps elements of a source specification into elements of a target specification

$$S \overset{TS}{\longmapsto} S'.$$

Concluding, this second part of specialization 2 can be characterized as factoring the specifications out of the other design state knowledge

$$DSK = LS \cup S \cup DSK_{other}$$

and a further partitioning of the state transitions

$$D = GDS \cup TS \cup D_{other}$$

Specialization 3 Our final effort must introduce still more reality into a model of the design process i.e, any knowledge which has a bearing on the assessment of rationality must be incorporated into the model. Since so called real-world design is resource constrained and supposedly guided by methods, techniques, heuristics, etc., this knowledge must be made an explicit part of the design state. In addition, we should like to ground the notion of routine design decision. Intuitively, routine connotes mechanizable and therefore, completely describable or equivalently, subject to a closed-world assumption. Thus, orthogonal to our final specialization which includes representations for resource and methodological knowledge as part of the design state, we also describe distinguished transitions for routine and nonroutine decision.

In our final specialization then, methodological knowledge is presumed to be expressable in terms of a linguistic system for design decision control knowledge, LS_{DCK}, and is explicitly recorded in a specification, S_{DCK}. (Section 4.2 gives the

grammar for LS_{DCK}.) Each of these are added as components of the design state. Similarly, resources are presumed to be definable in terms of one or more linguistic systems LS_R and explicitly represented as specifications S_R. In order to distinguish artifact specifications and linguistic systems we use the subscript A as in LS_A and S_A. Thus, our closed-world design state includes at least

$$\langle\langle LS_A, S_A\rangle, \langle LS_R, S_R\rangle, \langle LS_{DCK}, S_{DCK}\rangle\rangle$$

Other components will be added after we introduce our notion of routine decision.

Routine decisions are distinguished transform steps i.e., $RD \subset TS$. A routine decision is given by

$$\langle S_A, S_R\rangle \overset{RD}{\longmapsto} \langle S_A, S_R\rangle' \tag{4.4}$$

Thus a routine decision must concurrently change both an artifact specification and a resource specification. The change in the resource specification is associated with the cost of the routine decision. The change in the artifact specification is associated with the benefit of the routine decision. Thus, every routine design decision has an attendant cost and benefit each of which is presumed to be completely described by the mapping. Clearly this is an important part of any basis for classifying rationality.

Note that a change in resource specification without the attendant change in artifact specification is considered nonroutine. That is the nonroutine decision, NRD

$$S_R \times \mu(S_R) \overset{NRD_{\mu(S_R)}}{\longmapsto} S'_R$$

which is read as update of some resource (e.g., increase budget or decrease time) is not to be considered equivalent to

$$S_A \times S_R \overset{RD}{\longmapsto} S'_A \times S'_R$$

but where $S_A = S'_A$. This is interpreted as the routine decision which expends resources but effects no discernable change in the artifact specification.

In the interests of a comprehensive model of design decision rationality we need to enrich our closed-world design state, give further detail to our definition of routine decision in light of this enriched description, and define certain nonroutine decisions whose consequences have an impact on our assessment of rationality.

In addition to the nonroutine decision which effects change to a resource specification, we also identify the following prototypical nonroutine decisions:

- $NRD_{\mu(S_{DCK})}$ which modifies a specification of decision control knowledge

  $$S_{DCK} \times \mu(S_{DCK}) \overset{NRD_{DCK}}{\longmapsto} S'_{DCK}$$

  where $\mu(S_{DCK})$ is the creation, update, or deletion of some piece of decision control knowledge.

- $NRD_{\mu(LS)}$ which changes some part of a linguistic system

$$LS \times \mu(LS) \overset{NRD_{\mu(LS)}}{\longmapsto} LS'$$

where $\mu(LS)$ is the creation, update, or deletion of some piece of the linguistic system e.g., a new grammar rule, a change to a rule of inference for some semantic, or the deletion of an extra-logical symbol.

- NRD_{ψ} which designates the target linguistic system T (that subsequent routine decisions map to) from a collection of known linguistic systems

$$\psi(\{LS\}) \overset{NRD_{\psi}}{\longmapsto} T$$

- $NRD_{\mu(Lib(LS_i,LS_j))}$ which updates the library of mappings between two linguistic systems

$$Lib(LS_i, LS_j) \times \mu(Lib(LS_i, LS_j)) \overset{NRD_{\mu(Lib(LS_i,LS_j))}}{\longmapsto} Lib(LS_i, LS_j)'$$

When $i = j$ the mapping is called transform or an *intra*-linguistic system mapping. When $i \neq j$ the mapping is called refine or an *inter*-linguistic system mapping.

Thus, these nonroutine decisions are but particular characterizations of other kinds of design decisions D_{other} identified in 4.3 above. That is, $NRD_i \in D_{other}$ where i is one of the prototypes identified above.

In our final specialization leading to a view of the design process as a closed-world state transition system, the following three specializations are made. Firstly, the design state DSK_{other} is specialized to several specific components becoming the routine design state, RDS, with the following structure.

$\langle LS_{C,A}, S_{C,A} \rangle$ current base artifact linguistic system and specification

T_A target artifact linguistic system

$\langle LS_{C,R}, S_{C,R} \rangle$ current base resource linguistic system and specification

T_R target resource linguistic system

$Lib(LS_{C,A}, LS_{T,A})$ library of mappings from current base to and target artifact linguistic system

M bound mappings applicable to and the current base artifact specification

$\{LS\}$ the set of candidate target linguistic systems

Since we impose a closed-world assumption we need not include DSK_{other} or equivalently, $DSK_{other} = \emptyset$.

The function, *Lib*, taking two artifact linguistic systems as arguments produces a set of mappings

$$Lib(LS_{C,A}, LS_{T,A}) \rightarrow M$$

where M is the set of mappings which are applicable in the current state i.e.,

$$M = \{m | m \in Lib(LS_{C,A}, LS_{T,A}) \wedge applicable(m, S_{C,A})\}$$

Secondly, the transform step TS is specialized as in 4.4 above. A mapping induces a term in the language of the current base linguistic system for an artifact and resource to be changed into a term in the target linguistic system for the artifact and resource

$$m : L(LS_{C,A}) \times L(LS_{C,R}) \rightarrow L(LS_{T,A}) \times L(LS_{T,R})$$

Thirdly, the other design decisions D_{other} are restricted to the prototypical nonroutine decisions identified above i.e.,

$$D_{other} = \{NRD_{\mu(S_{DCK})}, NRD_{\mu(LS)}, NRD_{\psi}, NRD_{\mu(Lib(LS_i, LS_j))}\}$$

Thus we has satisfied the closed-world assumption from both a state and a transition viewpoint.

A routine design state transition is given by

$$RDS \times DT \longmapsto RDS' \tag{4.5}$$

where DT is a state transition which is either a routine decision RD or a nonroutine decision NRD. Having made these projections we are now prepared to define a function, *rat*, which determines the rationality level for any routine decision

$$rat : RDS \times m \mapsto \text{rationality ordinal}$$

That is, given a routine design state and a designated mapping m, the predicate *rat* determines an ordinal of rationality on the following scale

1 Not Rational
2 Trivially Rational
3 Non-dominantly Rational
4 Conjunctively Rational
5 Compensatorily Rational
6 Consensually Rational
7 Unanimously Rational

The next section gives our knowledge representation scheme for S_{DCK}. This is followed by a brief discussion of some of the practical ramifications for the specification of S_{DCK}. We then give procedure for the function *rat* which, using S_{DCK}, classifies routine decisions.

4.2 Representing Decision Control Knowledge

The decision control specification (S_{DCK}) is intended to be a particular operationalization of the generic design goal. See Figure 4.2. We consider this goal to be the root of a hierarchy of objectives, criteria, and attributes. Each objective expresses something to be pursued in the fulfillment of the design goal. Since there may be many objectives we have the potential for conflict among them. Each objective is further decomposed into criteria. Each criterion expresses some relevant aspect of either its superior objective or criterion. Each criterion is decomposed into subordinate criteria or attributes (leaf criteria). A weight is assigned to each criterion and attribute to indicate the relative salience of the node to its immediate superior. For each attribute a corresponding value function and observation channel are specified. The value function maps observation channel values onto some specified scale of desirability ranging from aversion through indifference to preference. The observation channel consists of a metric and a procedure for its application. The observation channel represents an objective measurement of either the artifact or resource specification. The value function turns this into a subjective measurement for purposes of decision control. A top-down perspective on the S_{DCK} indicates *how* the design goal and its subordinates are operationalized. The complementary bottom-up view indicates *why* a particular component is present i.e., the superior(s) which it serves.

More formally, the decision control knowledge specification, S_{DCK} consists of a nonempty set of objectives O. Each $o_i \in O$ is a tuple

$$\langle name, direction, C \rangle$$

where *name* is the label for o_i, $direction \in \{Maximize, Minimize\}$, and C is the objective's set of criteria. For example, Structured Design objectives O_{SD} might include:

$\langle ArtifactComplexity, Max, \{ModuleComplexity, InterfaceComplexity\}\rangle$
$\langle TransformationCost, Min, \{SentenceCount\}\rangle$

Each criterion $c_i \in C$ is a tuple

$$\langle name, weight, \{C \cup A\}^{+} \rangle$$

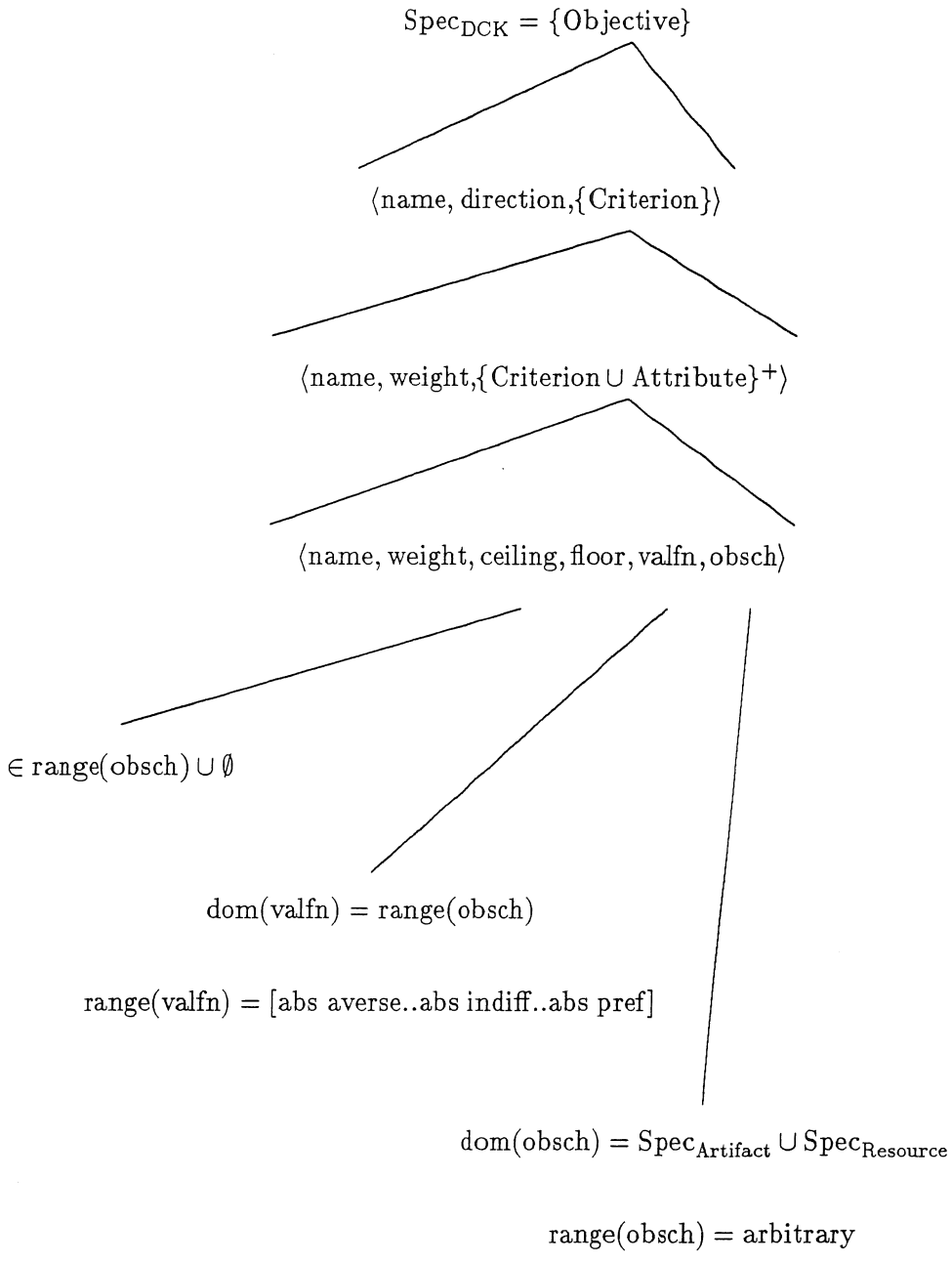

Figure 4.2: Representing Decision Control Knowledge.

where *name* is the name for c_i, $weight \rightarrow (0,1]$ indicates the proportional contribution of c_i to its superior, C is the criterion's set of subcriteria and A is the criterion's set of attributes. Either S or A may be empty but, $S \cup A$ must be nonempty. Continuing the Structured Design example, $C_{ArtifactComplexity}$ is

$$\begin{aligned} c_1 &= \langle ModuleComplexity, .4, \{Cyclomatic, Volume, Calls, Modules\}\rangle \\ c_2 &= \langle InterfaceComplexity, .4, \{Parameter, Globals\}\rangle \end{aligned}$$

By convention, the sum of all weights for all objectives is restricted to a small positive integer, usually 1.

Each attribute $a_i(x) \in A$ is a tuple

$$\langle name, weight, uc, lc, val(obs(x)), obs(x)\rangle$$

where *name* is the name for a_i, $weight \rightarrow (0,1]$ indicates the proportional contribution of a_i to its superior criterion, uc and lc are the upper and lower bound used for conjunctive cutoffs and must be $\in range(obs) \cup \emptyset$, $val(obs(x))$ is the attribute's value function, and $obs(x)$ its observation channel. Thus $A_{ArtifactComplexity}$ is

$$\begin{aligned} a_1(SD) &= \langle Cyclomatic, .1, \infty, 0, val_{Cyclomatic}, obs_{Cyclomatic}(SD)\rangle \\ a_2(SD) &= \langle Volume, .1, \infty, 0, val_{Volume}, obs_{Volume}(SD)\rangle \\ a_3(SD) &= \langle Calls, .1, \infty, 0, val_{Calls}, obs_{Calls}(SD)\rangle \\ a_4(SD) &= \langle Modules, .1, \infty, 0, val_{Modules}, obs_{Modules}(SD)\rangle \end{aligned}$$

where SD is a Structured Design specification i.e., either $S_{C,A}$ or $S_{T,A}$.

The $dom(obs)$ is RDS, but it is most likely confined to $S_{C,A}$, $S_{T,A}$, $S_{C,R}$ and $S_{T,R}$. By including both the current base and target specifications we allow observations to be defined over the change between transitions in addition to particular artifact or resource attribute values. The $range(obs)$ is arbitrary and may support total, partial, or no ordering. For example the observation channel $obs_{Cyclomatic}(SD)$ is defined as

$$\forall module \in SD, \sum cyclomatic(module)$$

where $cyclomatic(module)$ is as defined by [40].

Each value function maps its corresponding observation channel values onto any ordered set of values Usually, distinguished values representing

absolute aversion < absolute indifference < absolute preference

are specified. All other values, then, lie in

[absolute aversion, absolute indifference]

or in

[absolute indifference, absolute preference]

For example the reals in $[-1, +1]$ are quite often used where -1 corresponds to absolute aversion, 0 to absolute indifference, and $+1$ to absolute preference. In cases where such absolute preference is unknown, a simple transformation (quite often linear, though not necessarily always) is used. In any case, the $domain(val) = range(obs)$ and $range(val)$ is some arbitrary closed interval. Our Structured Design example a simple linear transformation of observation channel values such that $range(obs) = [0, \infty]$ with smaller values preferred over larger and we are indifferent between equal values.

This formulation of the S_{DCK} is presumed to represent all facets of control knowledge that a designer brings to bear in routine design decision-making. Any change to the S_{DCK} constitutes a non-routine decision.

4.3 Background and Guidelines for Components of the S_{DCK}

In every decision situation there are factual elements and value elements. The factual elements are those that can be verified independently. Value elements, on the other hand, defy independent verification—they are agent dependent. In the S_{DCK} we separate the factual from the valuational by distinguishing the notion of observation channel from the notion of preference valuation. The notion of an observation channel comes from [38]. The observation channel forms the essential input mechanism for encoding some perception of reality. Observation channels can be crisp or fuzzy (after the fuzzy set theory of [91]). Properties of the variable representing the observed phenomenon include ordering (none, partial, linear), whether or not distance is recognized, and continuity (discrete, continuous). Practically speaking each of Klir's formal prescriptions must be addressed in the synthesis of a demonstrably effective observation channel.

The value function component of the S_{DCK} serves to make a designer's preference explicit. A collection of value functions coupled with their sources constitutes a value system. Hence the S_{DCK} could be called the designer's value structure. Judgement, which signifies the act of giving an opinion, is the most common value element in any decision-making process. By separating the valuational element from the observation channel we can support the individual (and potentially different) preferences which may be ascribed to observed phenomena. In [82], von Wright gives a formal system for reasoning about preference. With this system, one has the ability

to define a notion of goodness and of badness. Moreover, these notions may be defined relative to an agent and an occasion. "What is good to one subject on some occasion may not be good to another subject on the same occasion or to the same subject on another occasion." Though a logical system for evaluating properties of preference expressions is not presently incorporated in our DDM, the decision analysis techniques used do presume that such properties as preference transitivity and consistency hold.

Scaling and measurement are fundamental to the successful modeling of an empirical system such as routine design decision making. It is through these that we are able to map some perception of reality into a model suitable for automated computation. The classification of scales is perhaps best considered in terms of the class of transformations on each scale which leave it invariant—those transformations which preserve the information it contains. In order of increasing strength the available scales are:

1. nominal - unique up to any 1-1 transformation (this consists essentially of assigning labels to objects).
2. ordinal- gives a rank order of objects and is invariant under monotone increasing transformations.
3. interval - unique up to positive linear transformations of the form $y = ax+b, a > 0$.
4. difference - invariant under a transformation of the form $y = x + b$.
5. ratio - invariant under positive linear transformations of the form $y = ax, a > 0$.

The essential difference between the ratio and interval scales is that whereas the ratio requires an origin as a point of reference, the interval scale does not. A formal treatment of scales of measurement is given in [11].

Scaling and measurement are essential activities in the specification of the observation channels and value functions. Separating the observation and value functions eases the difficult task of proceeding directly from physical performance scale values to preference scale values. Moreover, we can accommodate widely varying physical performance scales while mandating a common scale for preference.

The structuring of a hierarchy of objectives, criteria, and attributes is not an easy task. There is no set procedure, nor are decision makers, in general, and designers, in particular, likely to articulate these in the normal course of their tasks. Keeney and Raiffa conclude, "The intertwined processes of articulating objectives and identifying attributes are basically creative in nature." [37]. This supports our assertion that specifying the S_{DCK} is non-routine. Keeney and Raiffa also identify four desirable properties of a S_{DCK}:

1. completeness—all pertinent aspects of the problem are represented;
2. operational—can be utilized in some meaningful manner;
3. decomposable—relevant aspects of the problem can be disaggregated into their constituent parts; and
4. nonredundant—no aspect of the S_{DCK} overlaps any other.

To the extent that these properties can be met, we can claim that the S_{DCK} represents that knowledge which a designer uses in routine designing.

In conclusion, while there does not exist (and probably cannot exist) a sound meta-methodology for the specification of a S_{DCK}, a rich and useful variety of tools, techniques, methods and examples of their application is present in the literature. We now move to our formulation of the predicate, rational.

4.4 The Rationality of Routine Design Decisions

A unified approach to combining multiattribute decision methods is defined by Hwang and Yoon in [31]. Their basic idea is that multiple methods be applied in series and/or simultaneously instead of selecting a single "best" method for the situation. This idea coupled with our own observation that the different methods can be ordered as to their use of decision control knowledge form the basis for our assessment of rationality.

The purpose of Hwang and Yoon's initial screening is the elimination of dominated and infeasible alternatives and thereby a reduction in subsequent information processing requirements. Two non-compensatory techniques are used in this screening. Recall that they are deemed non-compensatory because each technique considers the valuation of an alternative on each objective, criterion, and attribute *independently*. No trade-off among objectives, criteria, or attributes is considered. The first technique is the dominance sieve. The second technique applied is the conjunctive cut-off method. Recall that this technique is known as *conjunctive* since all attributes must simultaneously achieve acceptable levels (cut-offs).

In Hwang and Yoons' next step four methods (linear assignment, simple additive weighting, TOPSIS, and ELECTRE) representing each of the types of compensatory techniques (scoring, compromising, concordance) are applied in parallel. The result is four potentially different orderings of the alternatives.

These four orderings are aggregated into a partially ordered set using three ranking techniques:

1. mean ranking is based on the average rank order over the four orderings;
2. majority rule ranking is based on a simple tally of the first- through fourth-place finishes in the four orderings;
3. wins-minus-losses ranking is based on the tally of the number of times each alternative was ranked above another (win) minus the number of times each alternative was ranked below another (loss).

Each ranking thus represents a different view of the ordering determined by independently applied compensatory methods. An aggregate ranking in the form of a partially ordered set is constructed from each of the linear rankings.

Hwang and Yoon conclude that the decision maker should choose that alternative which is aggregately ranked first or in the case of ties, one of the alternatives so ranked. This normative statement constitutes an action axiom. The definition of a formal decision method must contain at least one such prescriptive axiom for it defines the sufficient conditions for action [29]. A formal decision method is a conditional statement whose antecedent is the context of the decision and whose consequent is a recommendation for action. Since our purpose is to determine whether a designer has pursued (or is pursuing) a course of action consistent with a formal decision method, we must define the context of the decision and the action axiom(s). We have already defined the context of decision i.e., the state-transition model of design and the decision control knowledge.

The following definitions give six successively stronger action axioms.

Definition 4.1 (Trivially Rational) *A routine design decision is trivially rational if there exists but one mapping and this yields an improvement over the current design state for at least one attribute for the given* S_{DCK}.

Definition 4.2 (Non-dominantly Rational) *A routine design decision is non-dominantly rational if the consequence of the mapping yields a non-dominated alternative design state with respect to a given* S_{DCK}.

Definition 4.3 (Conjunctively Rational) *A routine design decision is conjunctively rational if it is non-dominantly rational and if the mapping yields an alternative design state which meets stipulated cut-offs for each attribute.*

Definition 4.4 (Compensatorily Rational) *A routine design decision is compensatorily rational if it is conjunctively rational and if the mapping yields an alternative design state which is ranked first by any of Linear Assignment or Simple Additive Weighting or TOPSIS or ELECTRE when evaluated against the same* S_{DCK}.

Definition 4.5 (Consensually Rational) *A routine design decision is consensually rational if it is compensatorily rational and if the mapping yields an alternative design state which is a superior element in the partially ordered set formed by joining three linear ranking—mean, majority rule, wins-minus-losses—of the disjunctive compensatory ordering.*

Definition 4.6 (Unanimously Rational) *A routine design decision is unanimously rational if it is compensatorily rational and if the mapping yields an alternative design state which is ranked first by each of Linear Assignment and Simple Additive Weighting and TOPSIS and ELECTRE when evaluated against the same* S_{DCK}.

Each of these definitions has been incorporated into our procedure for determining the rationality of routine design decisions (see Figure 4.3).

4.5 The Temporal Backdrop for Rationality

Since our rationality predicate uses the S_{DCK} and a set of alternative mappings as inputs, any change to these must constitute a change of routine design decision intervals. Thus the addition, modification, or deletion of an objective, criterion, attribute, value function, observation channel, or weight constitutes an interval boundary crossing. This is so because each of these changes can cause the predicate to return a different assessment.

We assume that transformation and refinement are the only way to change artifact specifications. Hence the libraries of transformations and refinements constitute a closed-world of alternative mappings. Thus, the addition, modification, or deletion of a transform or refinement will cause a change in the set of alternatives to be compared. This too can affect the outcome of the predicate.

Resource specifications are somewhat more difficult and potentially less amenable to the closed-world assumption. Thus, we assume that any change in a resource which is not directly caused by the deployment of a transformation or refinement is a non-routine decision.

We can now state that a routine decision interval begins with the selection and application of a transformation or refinement which only effects change in the artifact and resource specifications. The interval continues so long as the transformations and refinements effect the only change in the design state. The routine interval is terminated with a decision causing change to the S_{DCK}, the libraries of transformations or refinements, the linguistic systems, or the external imposition of resource changes.

Procedure 4.1 (Classify Rationality of Routine Design Decision) *Given: a specification of decision control knowledge, an observation vector for the current design state, a nonempty set of candidate mappings with associated observation vectors, and a designated choice from the set of candidates, determine the rationality of the designated choice.*

1. If the candidate set of mappings is not singleton, then proceed to Step 2. If the observation vector for the singleton mapping dominates the observation vector for the current design state, then terminate with the indication that the choice is trivially rational. Otherwise, terminate with the indication that the singleton mapping is dominated by the current state and is therefore, not rational.

2. Apply the dominance sieve to the design state associated with each mapping in the candidate set. If the designated choice is not a member of the non-dominated alternatives, then terminate with the indication that the designated choice is dominated and is therefore, not rational. Otherwise, the designated choice is at least non-dominantly rational.

3. If the non-dominated alternative set is singleton, then terminate with the indication that the designated choice is non-dominantly rational.

4. Apply the conjunctive cut-off sieve to the non-dominated alternative set.

5. If the designated choice is not a member of the conjunctively acceptable alternatives, then terminate the with the indication that designated choice is not conjunctively rational but is non-dominantly rational. Otherwise, the designated choice is at least both conjunctively rational and non-dominantly rational.

6. Apply the linear assignment method, the simple additive weighting method, the TOPSIS method, and the ELECTRE method to the conjunctively acceptable alternatives generating an ordering of alternatives for each method.

7. If the designated choice is ranked first in each of the four method's orders, then terminate with the indication that the designated choice is unanimously rational.

8. If the designated choice is not ranked first in at least one of the four preference orders, then terminate with the indication that the designated choice is conjunctively rational.

9. Apply the mean, majority rule, wins-minus-losses ranking techniques forming three sets of aggregate orders. From these construct the partially ordered set of aggregate preference.

10. If the designated choice is the highest ranking preference or one of the highest ranking in the case of ties, then terminate with the indication that the designated choice is consensually rational. Otherwise, terminate with the indication that the designated choice is compensatorily rational.

Figure 4.3: A procedure for determining the rationality of routine design decisions.

4.6 Conclusion

In this chapter we have integrated several notions from diverse disciplines to provide a model for the assessment of design decision rationality. Beginning with the LST state-transition model of the design process, we added the explicit representation of resources to the design state. We further noted that the LST model does not include the kind of knowledge necessary to justify *why* a particular mapping was selected. Hence, alone it cannot serve as a basis for rationality assessment. To address this we incorporated the general notion of decision control knowledge from decision theory.

However, two additional concepts were required. One concerned the separation of the act of observation (from general systems theory) from the act of valuation (from philosophy). The other concept dealt with the hierarchical decomposition of objectives, criteria, and attributes (a clear concession to the limits on human information processing). Use of a single decision analysis method was considered to be insufficiently comprehensive, especially given the variety of methods and their respective assumptions, limitations and viewpoints. Thus, we adopted an approach whereby several methods (each representative of a category of methods) are used. This, in turn, gives rise to a predicate which measures rationality on more than a binary basis.

Moreover by applying multiple decision analysis techniques, we gain a much finer granularity on the monotonicity of rationality for routine decision making. In particular for the non-compensatory techniques, monotonicity must be strictly increasing (decreasing) for equivalence classes of observed values for each attribute which is to be maximized (minimized). For compensatory techniques monotonicity is a function of preferred values. Interestingly though, since the techniques are compensatory, the monotonicity is based on the net increase (decrease) in preference (abhorrence). Finally, we set the whole system against a backdrop so that not only could routine decisions could be identified and their rationality determined, but also intervals of these decisions can be identified and distinguished from non-routine decisions. In conclusion, the integration of these concepts into a single system constitutes the principal contribution of this dissertation.

Chapter 5
Empirical Explorations

This chapter presents selected results from the application of our model. Using a well known problem from Structured Design, we illustrate each of the components of the S_{DCK} and we indicate how the rationality checking algorithm is used. A second example using extended entity-relationship design is also reviewed. We conclude with some observations on difficulties encountered and insights gained.

5.1 The Hospital Bed Monitoring Example

Significant difficulties were encountered applying our DDM to Structured Design (SD) in general and to this problem in particular. Most of the problems related to SD stem from it's largely informal basis, especially it's two principal design objectives, coupling and cohesion. Rather than redress this lack of formality, we chose as surrogates more formal criteria and metrics having some empirical basis for their claim that structural differences in program architectures do have an influence on software maintenance [22]. In addition to the problems in defining appropriate decision control knowledge, we also found the transformations to be even less well defined. This we redressed by first defining a grammar for SD. Then we redefined the informally described Structured Design transformations in terms of asserting, updating, and retracting sentences constituting a specification rendered in this grammar.

This particular problem from Structured Design was selected because of the attention it has received [77, 87, 66]. However, none of these authors gives a formal language definition for Structured Design descriptions nor can we find anything but informal discussions of transformations and refinements. Since we have mandated that the observation channels must be defined over at least some well-defined syntax (and preferably, a well-defined semantics), we were compelled to develop a BNF for Structured Design descriptions [56].

Our BNF goes beyond that required for Structure Charts. We have incorporated the regular expression grammar approach to data and procedure description [32, 83].

In addition, we do *not* require code for a module's procedural description. Rather, we emphasize specification of the relations typically left implicit or not even expressible in a Structure Chart language.

With this necessary grounding, we next hand translated Stevens' Hospital Bed Monitoring example [77]. Since no specification of each module's details was given, we had to reconstruct the module's internal relations from the structure chart alone. Interestingly, this process unearthed several errors and inconsistencies in Stevens' structure charts.

No formal description of structure chart transformations was given. Thus, we constructed a simple library of transformations from those implied by Stevens.

Finally, since no decision control structure was given, we postulated the following simple S_{DCK}.

5.2 A Simple S_{DCK} for Structured Design

In Figure 5.1 we have proposed a simple decision control structure for Structured Design. It consists of a two objectives: one is resource efficiency oriented while the other concerns artifact effectiveness.

The resource objective could be read as: "Minimize the amount of resource expended in applying a transformation as measured in terms of the number of sentences added, deleted, and updated in the design representation."

The artifact effectiveness objective is composed of two criteria. One criterion covers intra-module complexity while the other covers inter-module complexity. This is consistent with the perspective of [89]. Complexity due to modularity is described by four attributes: the cyclomatic complexity [40], the volume metric [26], the total number of module invocations, and the number of modules. Each of these well known metrics constitutes an observation channel and is implemented in the obvious manner. The interface complexity criterion has two attributes—one in which inter-module communication is effected via explicitly passed parameters and the other wherein some globally shared data module is used. Substantiation for the selection of these measures of Structured Design can be found in the empirical findings of [80, 36, 22].

Our assignment of weights is somewhat arbitrary. We initially decided that each attribute should contribute equally. However, owing to the restriction that all the weights should sum to one and that we only have nine attributes, we assigned a weight of .2 to the resource attribute and .1 to each of the artifact attributes.

Structured Design Decision Control Structure							
Objective	w	Criterion	w	Sub-Criterion	w	Attribute	w
Min Cost	.2	Transform	.2			# Sentences	.2
Min Complexity	.8	Module	.4			# Modules	.1
						$\sum$ Cyclo	.1
						$\sum$ Volum	.1
						# Calls	.1
		Interface	.4	Parameters	.2	# Parms	.1
						$\sum$ Size Parms	.1
				Globals	.2	# Globs	.1
						$\sum$ Size Globs	.1

The observation channels are:

Sentences - sentences changed by the transformation

Modules - total number of modules

$\sum$ Cyclo - sum of the cyclomatic complexity for each module

$\sum$ Volum - sum of the volume metric for each module

Calls - total number of module invocations

Parms - count of all parameters present at all interfaces

$\sum$ Size Parms - total size of all parameters present at all interfaces

Globs - total number of globals variables

$\sum$ Size Globs - total size of all global variables

Each value function is a simple linear transformation onto [0,1].

Figure 5.1: A decision control structure for Structured Design.

Given this structure, we classified each of Steven's decisions.

5.3 Classification of Stevens' Decisions

Strictly speaking each of Stevens' decisions is at best trivially rational since we are not informed of any alternative transformations for any decision. Notwithstanding the omission of alternatives, when our model is applied to Stevens' protocol (using the S_{DCK} previously described) the results are as summarized in Table 5.1.

This table is interpreted as follows. The column labeled 'Trn' indicates the decision or transition from state n to state $n+1$. Columns 2 through 11 are the observation channel values for each attribute. The column labeled 'Rat' indicates the rationality classification determined by our model. Thus the first triplet of rows shows:

1. the observation channel values for the initial state.
2. the change in observation channel values for the first decision. Thus for the expenditure of 13 changes to sentences in the artifact specification, one parameter of size 7 units is removed i.e., coupling was reduced by moving a call out of one module and into its parent. This decision is classified trivially rational.
3. the resulting observation channel values which now become the current state for the next decision.

Decision 4-5 is deemed not rational since the net change change in artifact attribute values is zero and it "cost" 5 units of work. The mapping turns out to be nothing more than a relabeling parameters.

Decision 6-7 factors a common function out of two modules replacing them with calls to the new module. Decisions 7-8 and 8-9 accomplish similar effects. That these decisions are classified as not rational suggests our S_{DCK} may be lacking in that it does not admit of what is perceived to be a desirable Structured Design objective i.e., a highly factored design. We probably need a counter balancing criterion for minimizing the number of modules. This example illustrates how our model might be used to elicit design decision control knowledge. A similar argument holds for decision 9-10 wherein a new level of abstraction is introduced to the specification with the introduction of a new module. Again the decision control knowledge should probably include an objective for maximizing some measure of specification understandability.

Finally, decision 13-14 induces significant change in the design specification with commensurate expenditure of resources. Here vector parameters are replaced with iterated calls passing an individual element of the iterated parameter. Besides the

reduction in coupling due to parameter passing, the additional beneficial side effect is a significant reduction in global coupling. For some unexplained reason Stevens withholds this mapping until the end of his discussion. If the alternative mappings were available at each decision point, as we have proposed, this decision would almost certainly have been taken much earlier. Quite possibly other decisions would have then been ruled out of contention. By not giving the alternatives considered at each state, we are condemned to guessing what motivated the design choices.

5.4 Rationality Assessment for Multiple Alternatives

Though treatment of Structure Chart transformation and refinement is largely informal (c.f. [52, 77, 89, 49]) we can at least generalize the transformations used in Steven's protocol. Thus in this section we demonstrate a detailed application of our rationality predicate when more than one mapping is applicable.

5.4.1 Observing Alternatives

Consider an initial observation matrix is shown in Table 5.2. The first column (Alt) identifies each alternative transformation. The remaining columns correspond to the attributes identified earlier. Note that the cost of changing the current state is zero. By convention we shall always reserve alternative 1 as the current state against which the states from the competing transformations are compared.

The alternative mappings in Table 5.2 are:

A1 `Current state`

A2 `Eliminate Funnel Module (Handle_Factors)`

A3 `Substitute Iterated Calls For Iterated Data Structure (Error_Flags, Error_Flags_2, Factors, Safe_Ranges, Unsafe_Flags)`

A4 `Factor Module (Write_Line_To_Stn) From (Notify_Stn_Of_Bad_Device, Notify_Stn_Of_Unsafe_Factors)`

A5 `Move Call To (Notify_Stn_Of_Unsafe_Factors) From (Handle_Factors) To (Monitor_Patients)`

A6 `Move Call To (Notify_Stn_Of_Bad_Dev e) From (Handle_Factors) To (Monitor_Patients)`

Trn	Count Sents	Count Mods	Sum Cyclo	Sum Volum	Count Calls	Count Parms	Size Parms	Count Globs	Size Globs	Rat
		12	28	878	11	38	336	8	8	
1-2	13	0	0	0	0	-1	-7	0	0	Triv
		12	28	878	11	37	329	8	8	
2-3	9	0	0	0	0	-1	-21	0	0	Triv
		12	28	878	11	36	308	8	8	
3-4	23	-1	0	0	-1	-5	-37	0	0	Triv
		11	28	878	10	31	271	8	8	
4-5	5	0	0	0	0	0	0	0	0	Not
		11	28	878	10	31	271	8	8	
5-6	21	0	0	0	0	-2	-14	0	0	Triv
		11	28	878	10	29	257	8	8	
6-7	10	+1	+1	+28	+2	+2	+2	0	0	Not
		12	29	906	12	31	259	8	8	
7-8	13	+1	+1	+24	+1	+3	+3	0	0	Not
		13	30	930	13	34	262	8	8	
8-9	16	+1	+1	+12	+1	0	0	0	0	Not
		14	31	942	14	34	262	8	8	
9-10	35	+1	+2	+32	+1	+3	+29	0	0	Not
		15	33	974	15	37	291	8	8	
10-11	8	0	0	0	0	0	0	0	0	Not
		15	33	974	15	37	291	8	8	
11-12	8	0	0	0	0	-2	+14	0	0	Not
		15	33	974	15	35	305	8	8	
12-13	1	0	0	0	0	0	0	0	0	Not
		15	33	974	15	35	305	8	8	
13-14	65	0	-6	-69	0	+2	-230	-7	-7	Triv
		15	27	905	15	37	75	1	1	

Table 5.1: The rationality for each decision in Stevens protocol.

Alt	Count Sent	Count Mod	Sum Cyclo	Sum Volum	Count Calls	Count Parms	Size Parms	Count Globs	Size Globs
A1	0	12	28	878	11	38	336	8	8
A2	52	11	27	850	9	33	299	8	8
A3	65	12	21	809	11	38	106	1	1
A4	24	13	30	889	13	39	417	8	8
A5	12	12	28	878	11	38	336	8	8
A6	12	12	28	878	11	38	336	8	8
A7	12	12	28	878	11	38	336	8	8

Table 5.2: The Hospital Bed Monitor initial observation matrix.

A7 `Move Call To (Store_Factors_In_Database) From (Handle_Factors) To (Monitor_Patients)`

In this example we have opted for value functions which are simple linear transformations of observed values. Thus the preference scale factor r for each cell is given by

$$r_{i,j} = x_j^{max} - x_{i,j} / x_j^{max} - x_j^{min}$$

where $i = 1 \ldots n$ for n alternatives, $j = 1 \ldots m$ for m attributes, x_j^{max}, x_j^{min} correspond to the maxima and minima for attribute j. One could argue that Structured Design proponents so disfavor global data references that this should be reflected in some value function which increases exponentially in the number of global references. Notwithstanding the work of numerous researchers in software metrics, we are unaware of any empirically validated or generally accepted Structured Design preference scheme, hence our admittedly conservative selection of value functions.

5.4.2 The Non-compensatory Techniques

Following Procedure 4.1 (see Figure 4.3), the first decision analysis method applied is dominance. The following dominance relationships were determined.

A1 is dominated by {}

A2 is dominated by {}

A3 is dominated by {}

A4 is dominated by {A1, A5, A6, A7}

A5 is dominated by {A1}

A6 is dominated by {A1}

A7 is dominated by {A1}

Note that alternatives A4, A5, A6 and A7 are dominated by the current state. That is, there is no benefit gained from the expenditure of resources for these transformations. In fact Stevens designates alternative A5 as his first choice for transformation. Our analysis indicates that it is not rational with respect to the given S_{DCK} since it yields a state which is inferior to the present state. In his defense however, alternate transformations A5, A6 and A7 are but pieces of the more macroscopic alternative transformation, A2. The only additional steps are the actual deletion of one, the call to `Handle_Factors` in `Monitor_Patients` and two, the module itself.

The next decision analysis technique used is the conjunctive cutoff. Here infeasible alternatives are ruled out because they fail to meet some minima or they

Alt	SAW	LAM	ELECTRE	TOPSIS
A1	3rd	3rd	2nd	1st
A2	2nd	2nd	1st	2nd
A3	1st	1st	3rd	4th
A4	4th	4th	4th	3rd

Table 5.3: The independent rankings of alternatives by SAM, LAM, ELECTRE, and TOPSIS.

exceed some maxima for each alternative. Consider for example that the number of modules should not exceed 12. Alternative A3 would no longer be feasible. We shall not impose this particular cutoff in order to carry sufficient alternatives into the next phase.

Let us assume that the designer did not designate alternative A5 and that alternative A3 is not eliminated by failing to meet a conjunctive cutoff. Thus, the set of non-dominated conjunctively acceptable alternatives feeding into the next phase are {A1, A2, A3, A4}.

5.4.3 The Compensatory Decision Analysis Techniques

Independent application of the four compensatory decision analysis techniques yields the rankings given in Table 5.3. If alternative A4 was the designated choice, the rationality algorithm would terminate at this point with an indication that it is conjunctively rational. That is, unlike alternatives A2 and A3 (A1 not being designated), alternative A4 is not ranked first by at least one of the techniques.

5.4.4 The Consensual Evaluation

Aggregation of the four preference orderings (SAM, LAM, ELECTRE, and TOPSIS) by the three ranking techniques yields Table 5.4. Thus in the aggregate partial ordering, alternative A4 is always less preferable to any of the other three. If it were the designated choice, the algorithm would indicate that alternative A4 is compensatorily rational (but not consensually rational). However, since alternative A2 is the single consensually rational choice, alternatives A1 and A3 are relegated to the classification of compensatorily rational.

This concludes our Structured Design example. In the following section we examine the application of DDM to conceptual data modeling.

Alt	mean ranking	majority rule	wins-minus losses
A1	(2,3) tie	(2,3) tie	3rd
A2	1st	1st	1st
A3	(2,3) tie	(2,3) tie	2nd
A4	4th	4th	4th

Table 5.4: The aggregation of the SAM, LAM, ELECTRE, and TOPSIS orderings.

5.5 Summary of Results for Conceptual Data Modeling

In [57] we give protocols for each of two examples of design decisions in conceptual data modeling. The underlying representations manipulated consist of Extended Entity-Relationship (EER) models [78]. In the following sections we give a decision control knowledge specification derived for conceptual data modeling. We summarize the transformations used to generate alternative schemata. The decision control knowledge specification and transformations are used in each of the following two sections for the examination of the rationality alternative conceptual data models.

5.5.1 S_{DCK} for Conceptual Data Modeling

Batini and Di Battista apply the concepts of coupling and cohesion when partitioning a Conceptual data model [6]. A Conceptual data model consists of a hierarchically related set of subschemas. Batini and Di Battista assert that in a "good" conceptual data model there exists a high degree of cohesion between concepts in the same subschema and a low degree of coupling between concepts in different subschemas.

Given their prescriptions, we have operationalized their rather informal goals into the simple decision control knowledge structur given in Figure 5.2

5.5.2 The Bottom-up and Top-down EER Transformations

The process of conceptual data modeling consists of the application of top-down and bottom-up transformations in synthesizing an EER model. The bottom-up transformations create concrete (or ground) concepts in the EER. The top-down

Conceptual Data Modeling Decision Control Structure					
Objective	w	Criterion	w	Attribute	w
Min Cost	.2	Transform	.2	# Transforms	.2
Min Complexity	.6	Coupling	.4	# Direct Relations	.2
		Partioning	.2	# Indirect Relation	.2
				# Partitions	.2
Max Cohesion	.2			Entities/Partition	.1
				Relations/Partition	.1

The observation channels are:

Transforms - count of the number of transforms applied;

Direct Relations - count of the number of direct relationships between partitions;

Indirect Relations is a count of the number of indirect relationships between partitions (i.e., similar to the notion of transitive relationships engendered by global coupling);

$\overline{Entities/Partition}$ - the average number of entities per partition; and

$\overline{Relations/Partition}$] - the average number of relationships per partition.

Each value function is a simple linear transformation onto [0,1].

Figure 5.2: A decision control structure for Conceptual Data Modeling.

transformations create abstract concepts in the EER and map these to other concrete and abstract concepts.

The bottom-up transformations are:

```
B1  createEntity: entityName
B2  createRelationship: relationshipName
        with: setOfEntities
B3  createGeneralization: generalizationName
        with: setOfEntities
B4  collect: setOfAttributes under: entityName
```

The top-down transformations are:

```
T1  mapEntity: entityName toRelationship: relationshipName
        with: setOfEntities
T2  mapEntity: entityName toGeneralizationOf: setOfEntities
T3  mapEntity: entityName toCollection: setOfEntities
T4  mapRelationship: relationshipName
        toParallel: setOfRelationships
T5  mapRelationship: relationshipName toEntity: entityName
        with: setOfRelationships
T6 elaborate: entityName with: setOfAttributes
```

Both transformation types are used to define the mapping between concrete and abstract EER schemata.

5.5.3 Example #1

Working from an initial concrete schema, Batini and Di Battista present three alternative mappings to abstract schemata. We have added a fourth. The observation matrix shown in Table 5.5 summarizes the performance of each alternative against the decision control knowledge attributes identified in Section 5.5.1. See [57] for the detailed transformations comprising each alternative abstract schema.

Thus, the alternative 1 (A1) observation vector summarizes the following facts. The number of transforms is 13 i.e.,

- four `createEntity` transforms, one for each abstract entity defining a partition;
- five `createRelationship` transforms, one for each relationship connecting abstract and/or concrete entities; and
- four `mapEntity` transforms, one for each projection of the abstract schema onto the concrete schema.

Alt	Transforms	Direct Relatn	Indirect Relatn	Parts	Avg E /Part	Avg R /Part
A1	13	4	3	4	2.25	1.00
A2	14	1	4	3	2.66	2.00
A3	12	0	7	3	2.66	1.33
A4	12	3	1	3	3.00	2.33

Table 5.5: Example #1 observation matrix.

The number of direct and indirect relationships result from the projections of abstract to concrete relationships. The direct relationships are analogous to parameters in an interface between modules. The indirect relationships can be though of as global connections.

The number of partitions is determined by the number of abstract entity types created in the partitioning step.

Finally, the average number of entity types per partition is calculated by dividing the number of partitions into the total of both concrete and abstract entity types in the schema. The calculation is similar for the average number of relationships per partition.

Thus, the observation matrix summarizes what we know about each alternative as perceived through each attribute.

Each of the alternatives in non-dominated. Therefore, the selection of any one is at least non-dominantly rational. Since no conjunctive cutoffs are specified, we shall assume that each alternative is also conjunctively acceptable. Moreover, since Batini and Di Battista did not indicate which alternative to implement we shall classify all alternatives. Independent application of the four compensatory decision analysis techniques yield the same ranking of alternatives:

$$A4 > A3 > A2 > A1$$

Thus A4 is classified consensually rational while each of the other alternatives is classified conjunctively rational. Note that the other alternatives (A1, A2, A3) are not classified compensatorily rational since none of them was ranked first by any of the compensatory decision analysis techniques.

5.5.4 Example #2

Since this example only considers two alternatives we give the bound transformations describing how each is defined. First, however, we give the bottom-up

transformations describing the initial concrete schema. After detailing the two alternatives, we give the observation matrix summarizing each alternative. We conclude with the rationality classification for each alternative.

The Initial Concrete Schema

The following bottom-up transformations define the initial schema corresponding to [6] (Figure 17b).

```
createEntity: PERSON
createEntity: TEMPORARY-PERSON
createEntity: RESIDENT-PERSON
createGeneralization: G1 forEntity: PERSON
    with: {TEMPORARY-PERSON, RESIDENT-PERSON}
createEntity: GROUP
createEntity: COHABITATION
createEntity: FAMILY
createGeneralization: G2 forEntity: GROUP
    with: {COHABITATION, FAMILY}
createEntity: AGGREGATE-GEOGRAPHIC-REFERENCE
createEntity: ELEMENTARY-GEOGRAPHIC-REFERENCE
createEntity: LODGING

createRelationship: R1   with: {FAMILY, LODGING}
createRelationship: R2.1 with: {PERSON, COHABITATION}
createRelationship: R2.2 with: {PERSON, FAMILY}
createRelationship: R3
    with: {RESIDENT-PERSON, AGGREGATE-GEOGRAPHIC-REFERENCE}
createRelationship: R4
    with: {GROUP, ELEMENTARY-GEOGRAPHIC-REFERENCE}
```

First Alternative Abstract Schema

The following transformations synthesize the first alternative abstract schema [6] (Figure 17a) by partitioning the concrete schema (Figure 17b) and constructing the set of top-down transformations mapping from the abstract schema to the concrete schema. (For the sake of quick visual reference, the name of each abstract entity ends with an '*'.)

```
Partitioning step
```

```
createEntity: GROUP*
createEntity: PERSON*
createEntity: GEOGRAPHIC-REFERENCE*

createRelationship: R1
    with: {GROUP*, LODGING}
createRelationship: R2
    with: {GROUP*, PERSON*}
createRelationship: R3
    with: {PERSON*, GEOGRAPHIC-REFERENCE*}
createRelationship: R4
    with: {GEOGRAPHIC-REFERENCE*, GROUP*}

Map abstract to concrete step

mapEntity: PERSON*
    toCollection: {PERSON, TEMPORARY-PERSON, RESIDENT-PERSON}
mapEntity: GROUP*
    toCollection: {GROUP, COHABITATION, FAMILY}
mapEntity: GEOGRAPHIC-REFERENCE* toRelationship: R5
    with: {AGGREGATE-GEOGRAPHIC-REFERENCE,
           ELEMENTARY-GEOGRAPHIC-REFERENCE}
mapRelationship: R2 toParallel: {R2.1, R2.2}
```

Second Alternative Abstract Schema

The second alternative abstract schema [6] (Figure 20) is synthesized by partitioning the concrete schema (Figure 13a) and constructing the set of bottom-up and top-down transformations mapping from the abstract schema to the concrete schema.

```
Partitioning step

createEntity: PERSON*
createEntity: GEOGRAPHIC-REFERENCE*
createEntity: PERSON-IN-FAMILY subsetOf: PERSON
createEntity: RESIDENT-PERSON subsetOf: PERSON
createEntity: FAMILY

createRelationship: R2'
    with: {FAMILY, PERSON-IN-FAMILY}
createRelationship: R3'
    with: {RESIDENT-PERSON, AGGREGATE-GEOGRAPHIC-REFERENCE}
createRelationship: R4'
```

Alt	Transforms	Direct Relatn	Indirect Relatn	Parts	Avg E /Part	Avg R /Part
A1	11	4	0	3	2.66	.33
A2	11	1	3	2	2.50	.50

Table 5.6: Example #2 observation matrix.

```
    with: {ELEMENTARY-GEOGRAPHIC-REFERENCE, FAMILY}
createRelationship: WAS-RESIDENT
    with: {AGGREGATE-GEOGRAPHIC-REFERENCE, FAMILY}

Map abstract to concrete step

mapEntity: PERSON*
    toCollection: {PERSON, RESIDENT-PERSON, PERSON-IN-FAMILY}
mapEntity: GEOGRAPHIC-REFERENCE* toRelationship: R5
    with: {AGGREGATE-GEOGRAPHIC-REFERENCE,
           ELEMENTARY-GEOGRAPHIC-REFERENCE}
```

The Observation Matrix for Example #2

The observation matrix shown in Table 5.5 summarizes the attribute values for each of the two alternatives defined in the previous sections.

Each alternative is non-dominated and conjunctively acceptable (since no cutoffs are given). Like Example #1 each of the compensatory techniques ranks A2 over A1. Thus, A2 is consensually rational while A1 is at best conjunctively rational. If we were to use the rationality classification normatively, we would recommend A2.

5.6 Conclusion

The application of our model in these two cases has offered some insight into the practicality of assessing designer performance. Several general classes of difficulty can be identified. Progress is seriously impeded by the lack of a definition of the linguistic system(s) governing the artifact and resource representations. When coupled with informal and incomplete description of transforms and refinements, much preparatory work must be accomplished to meet the prerequisites of our model. Finally, the paucity of clearly articulated S_{DCK} components and the unavailability of the original designer, condemns one to merely approximate what might have been the case in the

original design effort. Clearly, if one had our model incorporated as a component of a software development environment, then the capture of this essential prerequisite information would be greatly improved and could probably be made a natural by-product of the design effort.

Chapter 6
Conclusions

We conclude this dissertation by reiterating the features of our model. This is followed by a comparison with other approaches to design decision modeling. Finally we indicate some future research directions.

6.1 Unique Features of DDM

Every model is ultimately the expression of something we hope to understand in terms of things we think we do understand. The chain linking concepts in the model may be quite lengthy but must be grounded with some collection of primitives that we accept without question. Our model is grounded in decision theory and the LST generic design step model for software development.

We set out to establish a formal framework to answer the central design rationalization query,

> To what extent has a designer, on a particular occasion, using an explicit definition of 'good', decided rationally?

This effort entailed reformulating the query by giving formal definitions for each of its constituents.

The first step grounded the object of design—an artifact—and the process by which it came to be with explicit specifications. We adopted a state-transition basis for the software process consistent with a transformational viewpoint. Artifact and resource specifications are considered part of the state at any particular decision point in the design process. Moreover, each of these consists of well-formed sentences from some explicitly defined linguistic system. The linguistic systems too are considered part of the state since changes to them can have profound effects on that which may be observed as part of the decision analysis.

We introduced the specification of decision control knowledge as the basis for bridging between a designer's informal notion of "good" and the formal methods for analyzing decisions. Here we carefully delineated the objective (agent independent) means for observing properties of the artifact or resource specifications from the means for expressing the subjective (agent preference-based) means for expressing worth.

We gave a formal definition for routine decision and defined an algorithm for classifying the rationality of this class of decision.

Finally, given that there can be no absolute rationality, only relative, and given that no single decision analysis method comprehensively assesses all decision settings, we defined our notion of rationality based on the knowledge used in the various decision analysis techniques. Thus, our principal contribution is the specification of a closed interval of relative rationality. This ordinal scale for the rationality of routine design decisions coupled with a formal definition of routine design decision fulfills our requirement for unambiguously answering the DR query.

The significance of our contribution is summarized as follows. When the DR query is answered in the affirmative (i.e., for a particular designer, occasion, and artifact) and we are dissatisfied with one or more characteristics of the resulting artifact, then we know that the source of the dissatisfaction must lie in the decision control knowledge and/or the transformations used to produce the artifact. This conclusion logically follows since the DDM has indicated that the designer is not at fault insofar as rationality is concerned. Moreover, given the persistent, comprehensive design history, we have significant documentation to assist us in tracing the source of the offending characteristic. We are aware of no other system supporting such activities in the development of software.

However, the DDM does require significant commitments before yielding its answer: the articulation of decision control knowledge in the form of objectives, criteria, attributes, value (preference) functions, observation channels; formal specifications and their defining linguistic systems; and the specification of transformations defined over those specifications. In other decision settings, this pre-structuring imposed on practice has had significant affect on the decision makers and the quality of the object of their decisions [42]. We would expect similar results for the practice of software design. Finally, these commitments appear to be on the agenda for software engineering research [70] and should find their way into practice in the future.

6.2 Other Models of Design Decision

We encountered few researchers who focus exclusively on design decision making [86, 73, 74, 23]. Other researchers either ignore the topic [47] or subsume it under work on automated assistants [64].

A model of decision making for detailed design is given in [86]. In addition to the model an associated automated tool exists "for tracking the decision process during development so that relevant information (particularly assumptions) can be recorded and later retrieved to assist with system comprehension prior to a maintenance operation."

The overall structure of the model is that of a bipartite decision tree: concept nodes represent decision points and refinement nodes represent alternative implementations. Concept nodes may have properties attached which serve as the basis for deciding between alternative refinements. They represent individual characteristics considered important in comparing alternatives.

These properties are combined into a single objective function for each concept node. "In reaching a decision to select a particular refinement and reject others, a designer is essentially optimizing the objective function across the alternatives."

White describes a process wherein multiple paths through a forest of these decision trees (each with its own state) is explored interactively. The exploration proceeds until the designer commits to a particular path. No attempt is made to automate this exploration other than to provide the human designer with the results of some symbolic manipulation.

In contrasting White's model with our DDM we note the following. White relies on a single objective function to be optimized for each node. He does not indicate that any decision variable normalization is necessary. He implicitly adopts a linear rate of substitution as well as monotonicity for terms in the objective function. In essence he uses only one decision analysis technique—something approximating SAW (though this is not formally defined). He requires all designs to be expressed in a single design/programming notation. He completely ignores the role of process resources. Finally, White's purpose is "tracking the decision process so that assumptions surrounding a decision are recorded and, thus, available for system comprehension."

Sintzoff proposes "to express program designs by hierarchical specification of design decisions" [73, 74]. This is essentially a transformational basis for a design history. Though one could envision other kinds of knowledge represented in Sintzoff's decision frames, he only illustrates artifact knowledge. The frame for a decision consists of: a label identifying the decision being described; an antecedent giving

the preconditions necessary for application of the decision; a consequent stating the effects of the decision; a composition detailing the subdecisions (each described with a similar decision frame). Thus, Sintzoff sketches a metalanguage for the description of metaprograms—a second-order program (method) yielding other programs.

Sintzoff argues that the hierarchical composition of metaprograms is essential for "distinguishing overall objectives from auxiliary technicalities, strategies from tactics." Moreover, this composition should be expressed using: and-nodes to express the combination of subdecisions, or-nodes to express alternatives between subdecisions, subtrees to indicate the refinement of subdecisions, and leaves each of which identify some known solution to a design problem. It is unclear how this representation technique meshes, if at all, with our own for the DCS.

Some key issues in understanding the design process are identified in [76]. They identify four styles of design derivation: 1) rationalized descriptions - the final artifact is given with sufficient supporting description to convince the reader of its correctness; 2) design sequences - a generally linear and idealized sequence of development steps leading to the final artifact is given; 3) design spaces - the combination of all possible sequences of derivation constituting the design space for the artifact is supplied; 4) problem behavior graphs - the design space supplemented with the exploration paths and the order of exploration. Our approach lies somewhere between design spaces and problem behavior graphs. We do not give the entire design space but do give the order of exploration as well as the non-functional knowledge used in cutting-off search.

Artifact effectiveness and process efficiency often are not explicitly distinguished. We have shown how they can be treated in a common framework, even with multiple conflicting objectives for each.

Similarly, so called "nonfunctional" goals are not explicitly represented in the specifications or the derivation systems. We have shown how they can be. Moreover, we explicitly show their interaction in the attainment of an artifact's design.

The omission of details, such as nonfunctional goal knowledge, is not entirely motivated by a desire to simplify the presentation. Rather, specification languages just do not represent the great abundance of design knowledge and, thus, do not allow us to reason with and about it. Here we have incrementally advanced the state of the art by demonstrating a way of integrating resource and artifact reasoning within a common framework. Like [69] we also hold that responsible designing not only delivers a specification for the construction of some artifact but also includes the derivation and rationale for that design.

Measuring progress toward goals is largely an open issue. A designer does not necessarily follow a monotonic path to fulfillment of a design goal. This can be due

to incomplete or incorrect knowledge. Each of these may be remedied by learning. Though we only include the effects of learning—non-routine decision-making—in our model, it is some small consolation that it is explicitly addressed at all. Additionally, we can deal with some of the nonmonotonicity with some decision analysis techniques. We have explicitly provided for the inherent tradeoff in dealing with goal conflict.

There are potentially a large number of languages involved for specifications, partially developed programs, and final results. We have founded our DDM on this notion in the form of a linguistic system governing what can be represented and, therefore, made subject to the DCS.

We have not given the definitive answer to the knowledge representation issues in design decision modeling. Some of these concerns are addressed in the following discussion.

6.3 Future Research

Further research in design decision modeling is easily divided into three categories: theoretical advances, empirical investigation, and practical application.

Theoretical advances in DDM include the specification of a formal language of DDM along with a methodology for its application. Essentially this language defines what can be said about the process and artifacts of design decision making. Definition of its formal semantics will require significant effort. We have given a basis for further work by formally defining the concepts involved. Borrowing from Sintzoff's idea of hierarchically composable decisions we could envision a hierarchy of intension/extension levels of decision modeling. Each interior node in the hierarchy is simultaneously the intension for its subordinates and the extension of its superior. Perhaps this approach could help overcome the rather simple routine, non-routine dichotomy. The effect of this approach is to expand the closed-world assumption to include more of the non-routine decisions, thereby routinizing them. This would entail specifying the decision control knowledge for when to change for example the library of transformations. Clearly, such meta-level knowledge specification is a challenging undertaking.

The methodological implications are equally difficult. A great deal of effort was expended in identifying a base representation language for Structured Design that was amenable to the decision analysis. The great lesson learned was to focus on the essential relations implied in the Structure Chart language not on the nature of the pseudo-code. Specification of a decision modeling methodology would likely be a significant undertaking.

The addition of further decision analysis techniques to the ordinal scale of rationality is clearly warranted. However, classification of decision analysis techniques with respect to this scale is no mean feat. It is not clear that the simple dimensions of knowledge representability in terms of compensatory versus non-compensatory techniques is sufficient.

A potentially fruitful line of research is the extension of our model to include so called fuzzy reasoning. This has application in the specification of observation channels and the assignment of a particular rationality descriptor to a routine decision interval or to all the routine intervals. The ascription of the label non-dominantly rational to a large sequence of routine decisions wherein only one is so described is clearly too conservative.

An area ripe for empirical investigation is the assumption that more rational decisions will lead to better designs. This line of study would pursue the distinction between decision and outcome. An outcome is a future state of the world that is valued relative to other possibilities. A good decision is an action we take that is logically consistent with the alternatives we perceive, the information we have, and the preferences we feel. In the uncertain world of designing, good decisions can lead to outcomes subsequently deemed as bad, and vice versa. We hold the position that the prestructuring of the design decision setting will substantially improve the resultant designs. It should be empirically determined whether the quality of the design stems from the prestructuring or a descriptive DDM turned normative.

Finally in the practical realm we should consider pursuit of automated design assistants imbued with prespecified decision control structures. That is, we could consider moving the DDM from being a purely descriptive model to being a normative one. However, this would require developers to change the way they approach designing. The whole idea of a normative model arises when we are not satisfied with our functioning. In view of the many easily demonstrated lapses in human decision-making that we can observe, who would want to rely on unaided judgement for a complex and important design problem?

Development of practical design decision assistants squarely confronts the issue of scale. While the software quality identified by [10, 41] consisting of three objectives, 11 criteria, and 25 attributes are specified in our knowledge representation, we do not yet have the empirical support for how these should be weighted, nor have adequate observation channels been defined (all are presently forced to a common 0 to 10 agent dependent scale) and preference functions have not even been considered. Considerable effort would be involved redressing these issues.

Organizations are increasingly turning to team designing. As presently envisioned, the DDM supports only a single designer. The reconciliation of multiple

potentially conflicting decision control structures emanating from multiple designers is a challenging thought.

In closing we reiterate the goal satisfied by this work: To combine the normative, descriptive, and practical viewpoints of decision analysis into a single logical framework for the analysis of software design decision making.

6.4 Postscriptum

Nay, lad! *Deciding*'s not your ploy,
For that's a risky game.
It's *making a decision*
That's your surest road to fame.

Decide means to take action,
And actions rock the boat,
And if you act and don't succeed,
Small chance you'll stay afloat.

But...making a decision,
Ah! that's the way to swing.
It keeps the masses happy
And doesn't change a thing.

So get yourself a task force
Well skilled in all the arts
And call them all together
And watch them flip their charts.

For Jack says no and Jim says yes
And Billy says perhaps
And Chester asks good questions
...When he isn't taking naps.

And Bertram, chomping his cigar,
Is chock full of statistics,
While Waldemar, who puffs a pipe,
Is famed for his heuristics.

"The figures prove—" "The model says—"
"The forecast bears me out."
"The complex simplex program
Shows I'm right without a doubt."

Let's tiptoe out and close the door
And let them stew a while.
No fear that they'll do something rash,
For *doing*'s not their style.

Reality's an untamed beast
That's difficult to master,
But models are quite docile
And give you answers faster.

So diddle with a model
To glorify your name,
Then get yourself a task force
And learn to play the game.

From *Decision Tables*
by M. Montalbano (SRA, 1974).

Bibliography

[1] ACKOFF, R. *The Art of Problem Solving.* John Wiley and Sons, Inc., New York, 1978.

[2] ARCHER, B. Design as a discipline. *Design Studies 1*, 1 (Jan. 1979), 17–20.

[3] ASIMOW, M. *Introduction to Design.* Prentice-Hall, Englewood Cliffs, NJ, 1962.

[4] BALZER, R. Automated enhancement of knowledge representations. In *IJCAI-85* (Aug. 1985), AAAI.

[5] BARTELY, W. Theories of rationality. In *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge.* Open Court, La Salle, Illinois, 1987, pp. 205–214.

[6] BATINI, C., AND BATTISTA, G. A methodology for conceptual documentation and maintenance. *Information Systems 13*, 3 (1988), 297–318.

[7] BELADY, L. Software engineer, the system designer. In *Software Engineering '86.* Peter Peregrinus, Ltd, London, 1986, pp. 421–425.

[8] BROOKS, F. *The Mythical Man-Month.* Addison-Wesley, Reading, MA, 1975.

[9] BROOKS, F. No silver bullet: Essence and accidents of software engineering. *IEEE Computer 20*, 4 (Apr. 1987), 10–19.

[10] CAVANO, J., AND MCCALL, J. A framework for the measurement of software quality. In *Software Quality Assurance Workshop* (Nov. 1978), ACM, pp. 133–139.

[11] CHANKONG, V., AND HAIMES, Y. *Multiobjective Decision Making.* North-Holland, New York, 1983.

[12] CHERNIAK, C. *Minimal Rationality.* MIT Press, Cambridge, MA, 1986.

[13] CROSS, N., NAUGHTON, J., AND WALKER, D. Design method and scientific method. *Design Studies 2*, 2 (Oct. 1981), 198–201.

[14] DASARATHY, B. Smart: Similarity measure anchored ranking technique for the analysis of multidimensional data analysis. *IEEE Transactions on Systems, Man, and Cybernetics SMC-6*, 10 (1976), 708–711.

[15] DIJKSTRA, E. *Selected Writings on Computing.* Springer-Verlag, New York, 1982.

[16] FREEMAN, P. Toward improved review of software design. In *National Computer Conference* (1975), ACM, pp. 329–334.

[17] FREEMAN, P. The central role of design in software engineering. In *Software Engineering Education.* Springer-Verlag, Amsterdam, 1976.

[18] FREEMAN, P. *Software Perspectives.* Addison Wesley, Reading, MA, 1987.

[19] FREEMAN, P., AND (EDS.), A. W. Fundamentals of design. In *Software Design Techniques (4th ed.).* IEEE Computer Society, Los Alamitos, CA, 1983.

[20] GERO, J. *Design Optimization.* Academic Press, New York, 1985.

[21] GERO, J. *Optimization in Computer-Aided Design.* North-Holland, Amsterdam, 1985.

[22] GIBSON, V., AND SENN, J. System structure and software maintenance. *Communications of the ACM 32,* 3 (Mar. 1989), 347–357.

[23] GILB, T. *Principles of Software Engineering Management.* Addison-Wesley, Reading, Massachusetts, 1988.

[24] GLEGG, G. *The Development of Design.* Cambridge Press, Cambridge, England, 1981.

[25] GRIES, D. *The Science of Programming.* Springer-Verlag, Amsterdam, 1981.

[26] HALSTEAD, M. *Elements of Software Science.* Elsevier North-Holland, New York, 1977.

[27] HAMILTON, M., AND ZELDIN, S. Higher order software – a methodology for defining software. *IEEE Transactions on Software Engineering SE-2,* 3 (Mar. 1976).

[28] HOARE, C. Programming: Sorcery or science? *IEEE Software 1,* 2 (Apr. 1984).

[29] HOLTZMAN, S. *Intelligent Decision Systems.* Addison-Wesley, New York, 1989.

[30] HWANG, C., AND MASUD, A. *Multiple Objective Decision Making - Methods and Applications.* Springer-Verlag, New York, 1979.

[31] HWANG, C., AND YOON, K. *Multiple Attribute Decision Making.* Springer-Verlag, New York, 1981.

[32] JACKSON, M. *Principles of Program Design.* Academic Press, New York, 1975.

[33] JONES, J. *Design Methods.* John Wiley and Sons, New York, 1980.

[34] JONES, J. *Essays in Design.* John Wiley and Sons, New York, 1984.

[35] JONES, J. *Programming Productivity.* McGraw-Hill, New York, 1986.

[36] KAFURA, D., AND HENRY, S. Software quality metrics based on interconnectivity. *Journal of Systems and Software 2* (1981), 121–131.

[37] KEENEY, R., AND RAIFFA, H. *Decisions with Multiple Objectives.* John Wiley & Sons, New York, 1976.

[38] KLIR, G. *Architecture of Systems Problem Solving.* Plenum Press, New York, 1985.

[39] MAYALL, W. *Principles in Design.* Design Council, London, 1979.

[40] MCCABE, T. A complexity measure. *IEEE Transactions on Software Engineering SE-2* (Dec. 1976), 308–320.

[41] MCCALL, RICHARDS, AND WALTERS. Factors in software quality, 1977.

[42] MILLER. *Professional Decision-Making.* Praeger Publishers, New York, 1970.

[43] MILLER, D., AND STARR, M. *The Structure of Human Decisions.* Prentice-Hall, Englewood Cliffs, NJ, 1967.

[44] MILLS, H. *Software Productivity.* Little, Brown and Co., Boston, 1983.

[45] M.M. LEHMAN, V. STENNING, W. T. Another look at software design methodology. *ACM Software Engineering Notes 9*, 2 (1984), 38–53.

[46] MOSTOW, J. Rutgers workshop on knowledge-based design. *SIGART Newsletter*, 90 (Oct. 1984), 19–32.

[47] MOSTOW, J. Toward better models of the design process. *The AI Magazine* (Spring 1985), 44–57.

[48] MYERS, G. *Reliable Software Through Composite Design.* Petrocelli Charter, New York, 1975.

[49] MYERS, G. *Composite Structured Design.* Van Nostrand, New York, 1978.

[50] NAUR, P., AND (EDS.), B. R. Software engineering: A report on a conference sponsored by the nato science committee, 1969.

[51] O'CATHAIN, C. Why is design logically impossible? *Design Studies 3*, 3 (July 1982).

[52] PAGE-JONES, M. *The Practical Guide to Structured Systems Design.* Yourdon Press, Englewood Cliffs, NJ, 1980.

[53] PARNAS, D. On the criteria to be used in decomposing a system into modules. *Communications of the ACM 15*, 12 (Dec. 1972).

[54] PARNAS, D., AND CLEMENTS, P. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering SE-12*, 2 (Feb. 1986), 251–257.

[55] PETROSKI, H. *To Engineer Is Human.* St. Martins Press, New York, 1985.

[56] PIDGEON, C. A Design Decision Protocol for Stevens' Hospital Bed Monitor Structured Design. Tech. Rep. ASE-RTP100, University of California, Irvine, 9 1989.

[57] PIDGEON, C. Two design decision protocols in conceptual data modeling. Tech. Rep. ASE-RTP099, University of California, Irvine, 9 1989.

[58] POPPER, K. *Conjectures and Refutations.* Routledge and Kegan Paul, London, 1963.

[59] POST, J. Paradox in critical rationalism and related theories. In *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge.* Open Court, La Salle, Illinois, 1987, pp. 225–278.

[60] PRESSMAN, R. *Software Engineering.* McGraw-Hill, New York, 1987.

[61] PROTZEN, J. Reflections on the fable of the caliph, the ten architects, and the philosopher. *Design Studies 3*, 2 (Apr. 1982), 85–91.

[62] PUGH, S. Design–the integrative-enveloping culture–not a third culture. *Design Studies 3*, 2 (Apr. 1982), 93–96.

[63] PYE, D. *The Nature and Aesthetics of Design.* Barrie and Jenkins, Ltd., London, 1978.

[64] RICH, C., AND WATERS, R. *Artificial Intelligence and Software Engineering.* Morgan Kaufman, Los Altos, CA, 1986.

[65] RITTLE, H. Some principles for the design of an education system for design. *Journal of Architectural Education XXVI*, 1-2 (Winter-Spring 1971).

[66] ROSTENSTREICH, S., AND HOWDEN, W. Two-dimensional program design. *IEEE Transactions on Software Engineering SE-12*, 3 (Mar. 1986), 377–384.

[67] RZEVSKI, G. On the design of a design methodology. In *Design: Science and Method* (1980), Design Research Society.

[68] SAATY, T. *The Analytic Hierarchy Process.* McGraw-Hill, New York, 1980.

[69] SCHERLIS, W., AND SCOTT, D. First steps towards inferential programming. In *IFIP Congress 83.* North-Holland, Amsterdam, 1983.

[70] SHAW, M. Beyond programming-in-the-large: The next challenges for software engineering. Tech. Rep. CMU/SEI-86-TM-6, Carnegie Mellon University/Software Engineering Institute, May 1986.

[71] SIMON, H. *Administrative Behavior.* The Macmillan Co., New York, 1961.

[72] SIMON, H. *Models of Discovery.* D. Reidel, Dordrecht, Holland, 1977.

[73] SINTZOFF, M. Suggestions for composing and specifying program design decisions. In *Lecture Notes in Computer Science, Vol. 83.* Springer-Verlag, New York, 1980, pp. 311–326.

[74] SINTZOFF, M. Understanding and expressing software construction. In *Program Transformation and Programming Environments.* Springer-Verlag, Berlin, 1984, pp. 169–180.

[75] STADLER, W. *Multicriteria Optimization in Engineering and in the Sciences.* Plenum Press, New York, 1988.

[76] STEIER, D., AND ANDERSON, P. Comparing algorithm syntheses, Mar. 1988.

[77] STEVENS, W. *Using Structured Design.* Wiley-Interscience, New York, 1981.

[78] T.J. TEOREY, G. WEI, D. B., AND KOENIG, J. Er model clustering as an aid for user communication and documentation in database design. *CACM 32*, 8 (Aug. 1989), 975–987.

[79] TRIBUS, M. *Rational Descriptions, Decisions, and Designs.* Pergamon Press, New York, 1969.

[80] TROY, D., AND ZWEBEN, S. Measuring the quality of structured designs. *Journal of Systems and Software 2* (1981), 113–120.

[81] TURSKI, W., AND MAIBAUM, T. *The Specification of Computer Programs.* Addison-Wesley, Reading, MA, 1987.

[82] VON WRIGHT, G. *The Logic of Preference.* Aldine Publishing Co., Chicago, IL, 1963.

[83] WARNIER, J. *Logical Construction of Programs.* Van Nostrand, New York, 1974.

[84] WEBSTER. *Webster's New Twentieth Century Dictionary.* The Publishers Guild, New York, 1967.

[85] WEINBERG, G. *Rethinking Systems Analysis and Design.* Little, Brown and Co., Boston, MA, 1982.

[86] WHITE, J. A decision tool for assisting with the comprehension of large software systems. In *Automated Tools for Information Systems Design.* North-Holland Publishing Co., New York, 1982, pp. 49–65.

[87] W.P. STEVENS, G. M., AND CONSTANTINE, L. Structured design. *IBM Systems Journal 13*, 2 (1974), 115–139.

[88] YOON, K. *Systems Selection by Multiple Attribute Decision Making.* PhD thesis, Kansas State University, 1980.

[89] YOURDON, E., AND CONSTANTINE, L. *Structured Design.* Prentice-Hall, Englewood Cliffs, NJ, 1979.

[90] YU, P. *Multiple-Criteria Decision Making.* Plenum Press, New York, 1985.

[91] ZADEH, L. Fuzzy sets. *Information and Control 8* (1965), 338–353.

[92] ZELENY, M. *Multiple-Criteria Decision Making.* Mc Graw-Hill, New York, 1972.