May 1987

Explaining and Refining Decision-Theoretic Choices

David A. Klein
University of Pennsylvania

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-87-57.

Explaining and Refining Decision-Theoretic Choices

Abstract

As the need to make complex choices among competing alternative actions is ubiquitous, the reasoning machinery of many intelligent systems will include an explicit model for making choices. Decision analysis is particularly useful for modelling such choices, and its potential use in intelligent systems motivates the construction of facilities for automatically explaining decision-theoretic choices and for helping users to incrementally refine the knowledge underlying them. The proposed thesis addresses the problem of providing such facilities. Specifically, we propose the construction of a domain-independent facility called UTIL, for explaining and refining a restricted but widely applicable decision-theoretic model called the *additive multi-attribute value model*. In this proposal we motivate the task, address the related issues, and present preliminary solutions in the context of examples from the domain of intelligent process control.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-87-57.

EXPLAINING AND REFINING DECISION-THERORETIC CHOICES

Dave A. Klein

MS-CIS-87-57
LINC LAB 74

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

June 1987

Acknowledgements: This work is supported, in part, by the NASA Graduate Student Researchers Programs and by DARPA grants NOOO14-85-K-0018, NSF-CER grant MCS-8219196 and U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027.

Explaining and Refining Decision-Theoretic Choices

Doctoral Thesis Proposal

David A. Klein*
LINC Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Technical Report MS-CIS-87-57 (LINC 74)

Abstract

As the need to make complex choices among competing alternative actions is ubiquitous, the reasoning machinery of many intelligent systems will include an explicit model for making choices. Decision analysis is particularly useful for modelling such choices, and its potential use in intelligent systems motivates the construction of facilities for automatically explaining decision-theoretic choices and for helping users to incrementally refine the knowledge underlying them. The proposed thesis addresses the problem of providing such facilities. Specifically, we propose the construction of a domain-independent facility called UTIL for explaining and refining a restricted but widely applicable decision-theoretic model called the *additive multiattribute value model*. In this proposal we motivate the task, address the related issues, and present preliminary solutions in the context of examples from the domain of intelligent process control.

Thesis Committee:
Dr. Norman I. Badler, University of Pennsylvania
Dr. Eric K. Clemons, The Wharton School
Dr. Timothy W. Finin, University of Pennsylvania
Dr. Aravind K. Joshi, University of Pennsylvania
Dr. Edward H. Shortliffe, Stanford University
Dr. Martin O. Weber, RWTH Aachen (Germany)

* This work is supported, in part, by the **NASA Graduate Student Researchers Program**

CONTENTS

Chapter I: Introduction

It's Saturday night. You think about going to a movie, a relatively inexpensive endeavor, but the ones you would like to see still involve waiting on long lines. You contemplate a nice dinner out, and that new Italian place probably won't be crowded, but you're over your credit limit. A Broadway show would be fun, and you can get cheap tickets at Duffy Square without waiting too long, but you hesitate to ride the New York subways at night. What will you do?

Suppose that you are a computer operator. You are leisurely sipping a soda when you notice that operating system queue space is almost exhausted. Scrambling to resolve the problem, you find that a huge dataset is waiting for a designated printer, but that the printer is currently disabled. You could just delete the dataset, but that would anger the affected user. You could print the dataset on a high-speed printer that only uses expensive paper, but that would waste resources. You think about copying the dataset to tape and printing it later when things have calmed down, but that's a lot of work for you and greatly increases the user's turnaround time. You contemplate sending the dataset back to the user's private disk storage, but that will require him[1] to send it back to the operating system later on and will likewise increase his turnaround time. You must do *something* because additional output is being generated on the queue and its complete exhaustion will crash the system. What will you do?

The need to choose among competing alternatives is ubiquitous in reasoning. We face judgement-intensive choices in all sorts of settings, from the mundane and unimportant (choosing an activity on a Saturday evening) to the highly technical and important (choosing an action to avoid a crisis in a computer installation). The field of inquiry concerned with addressing such choices in a formal, structured fashion is known as *decision analysis*, described by Ralph Keeney (1982) as 'a formalization of common sense for decision problems which are too complex for informal use of common sense'.

Our work rests upon the view that decision analysis provides a particularly desirable model for making complex choices among competing alternatives in intelligent systems.[2] The potential for employing decision-analytic models in intelligent system architectures motivates the construction of facilities for automatically explaining decision-theoretic choices and for helping users to incrementally refine the knowledge underlying them, as it is by now agreed that automated explanation and acquisition are important supporting capabilities for any intelligent system. The proposed thesis addresses the problem of providing such facilities.

Specifically, we propose to develop a system called UTIL[3] which takes an *additive multiattribute value function* (a restricted decision-theoretic model to be described later) and supplementary knowledge structures as input, and uses this information to interactively help the user to understand the justification for choices and to modify the underlying value function when such justifications are deemed unconvincing. We believe that the task of developing such a system is interesting, challenging, and tractable.

UTIL's successful completion would provide contributions to both artificial intelligence (AI) and decision analysis. From the perspective of AI, UTIL would extend previous work in automated explanation and knowledge acquisition, being among the first efforts to provide facilities for explaining and refining decision-theoretic choices. As the existence of such facilities would encourage

---

1 Masculine references such as 'him' are used throughout this document as a convenient alternative to references like 'him or her' and are not intended to connote gender.

2 We employ the general term *intelligent system* throughout this document rather than terms such as *expert system* or *decision support system* so as not to limit the discussion to particular architectures for computer-based decision-making.

3 *Util* (sometimes spelled *utile*) refers to a hypothesized unit of 'utility' or satisfaction.

the use of decision-analytic models in intelligent system architectures, UTIL would also provide a contribution to intelligent systems research. From the decision-analytic perspective, facilities for automatically justifying choices may play an important role in influencing the behavior of decision makers. Facilities for incrementally restructuring decision-theoretic models would address important open problems in decision analysis such as how to capture changing preferences over time and how to handle bias in decision-theoretic models. In addition, useful facilities for refining decision-theoretic models would help to reduce the demands on methods for acquiring such models from scratch, another active research area in decision analysis. These contributions are discussed in more detail in chapter VII.

While the proposed work may be said to lie at the crossroads of decision analysis and AI, this document is written assuming that the audience is familiar with AI and less familiar with decision theory. We apologize to the complement of this audience, who will find the expository sections on decision theory uninteresting (although we have tried to flag these in the text) and sections which tersely refer to AI architectures and well-known works in AI less than clear.

The document is organized as follows. Chapter II motivates the development of UTIL, largely consisting of arguments which support the employment of decision-theoretic models in intelligent systems. In chapter III, we provide a more explicit statement of our proposed goals and the research issues which arise in attempting to meet them. Next, we take a step back to review related foundational works, relevant previous efforts, and works which contrast with our proposed approach, in chapter IV. Chapter V is intended to communicate the flavor of the solutions we are seeking in response to the problems identified in chapter III. Chapter VI describes our research plans. Chapter VII reviews the potential contributions of the proposed work in detail.

Chapter II: Motivation and Problem Statement

The proposed research is motivated by *pragmatic* concerns regarding the usage of intelligent systems. The work focuses on the goal of making such systems more useful, rather than that of modelling cognition, exploring the nature of intelligence, or other goals commonly associated with artificial intelligence.

Specifically, we are interested in facilitating the development, operation, and maintenance of large intelligent systems that involve *complex choices among competing alternatives.* The process of choosing among alternatives may be the sole task of an intelligent system or a component task of some more encompassing reasoning framework.

In this chapter we argue for the use of decision theory as a mechanism for choosing among competing alternatives in intelligent systems, and motivate the development of sophisticated facilities for (i) explaining choices that are based on decision-theoretic models and for (ii) refining such models on an ongoing basis. The discussion proceeds as follows.

In section 1 we note that every intelligent system encompasses *some* paradigm for making choices, and we delineate the roles of implicit and explicit models of choice in such systems. In section 2 we build upon this distinction by describing an application which employs an implicit model where an explicit model would have been more appropriate, and expose the resulting difficulties. Having motivated interest in explicit models of choice, we take a step back in section 3 to examine some general characteristics which render such models suitable for (i) making competent choices, (ii) justifying these choices, and (iii) incremental modification. Next, in section 4, we introduce decision-theoretic models and describe a particular model which will provide the focus for the proposed research. In section 5 we examine these models in light of the desiderata of section 3, concluding that they provide desirable machinery for making complex, knowledge-based choices in intelligent systems. It follows that automated facilities for explaining and refining choices should be constructed, as we explain in section 6.

1. Explicit models of choice in intelligent systems

Before addressing the relative merits of using decision-theoretic models in intelligent systems, we take a broad look at the nature of choices in such systems and the models which support them.

1.1 Implicit and explicit models of choice

The reasoning machinery of any intelligent system constructed to perform any task in any domain encompasses -- *either implicitly or explicitly* -- one or more paradigms for making choices. Objects in intelligent systems (e.g., propositions to assert, rules to fire, subgoals to prove, program statements to execute, recommendations to display for users) may be chosen or ordered[4] according to one or more of the following methods:

1. A priori *ordering*: This describes, for example, procedures in traditional programming languages (in which statements are ordered for execution) and priority conflict resolution algorithms in production systems (in which rules are ordered for execution).
2. *Ordering according to a hard-wired (black box) algorithm*: Examples of this scheme include hard-wired conflict resolution algorithms (e.g., those employing special case, recency, distinctiveness rules (McDermott & Forgy 1978) or combinations thereof as in OPS5 (Forgy 1981)) and most heuristic evaluation functions in AI game-playing programs (Nilsson 1980).

[4] In this context, an *ordering* is simply the result of repeated choosing.

3. *Arbitrary ordering*[5] : This describes, for example, arbitrary selection rules in production systems (McDermott & Forgy 1978) and subgoal selection schemes in some logic programming languages.
4. *Ordering according to a coherent model of choice that is parameterized by domain knowledge*: Examples of this scheme include hand-crafted selection schemes in medical therapy planning systems (e.g., Clancey 1984) and production systems driven by decision-theoretic models (e.g., White & Sykes 1986).

If objects are ordered according to any of (1) through (3), we might say that the employed model of choice is *implicit*, in that the domain-specific factors underlying the choice (e.g., objectives, expressions of desirability) are not explicitly represented. In the case of (1), the justification for the ordering of objects remains outside the system, stored away in the programmer's mind. In (2), the basis for choices lies hidden in the code which implements the selection algorithm. In (3) there is effectively no identifiable knowledge driving choices.

If choices are made using some form of (4), we call the model of choice *explicit*. Elements of explicit models of choice include:

- There exists some natural and clear correspondence between the computational objects of selection (e.g., rules, procedures, values, logical assertions) and objects in the domain (e.g., therapies, dinner entrees, power plant recovery procedures).
- The factors driving choices (e.g., the need to maximize safety, the likelihood of allergic reaction) are explicitly represented.
- The factors driving choices are combined according to some theory of choice (e.g., utility theory, Bayes' theorem).

Explicit models may be encoded 'top-level' structures or as component structures of more complex models. In the former case, the model of choice is the principal reasoning machinery of the system and the sole mission of such a system is to help the user to choose between competing alternatives in some knowledge-intensive domain. In the latter case, where the model of choice is one component of an intelligent system which coexists with other structures, the process of choosing among alternatives works in cooperation with other distinct knowledge-based tasks such as generating or invoking those alternatives.

1.2 Roles for implicit and explicit models

We can examine the respective roles of implicit and explicit models of choice in terms of the general capabilities that intelligent systems are intended to support: competent reasoning, automated explanation, and automated acquisition. For models of choice, *competent reasoning* refers to the process of responsibly selecting among competing alternatives. *Explanation* involves generating a convincing justification for the choice of a particular alternative. *Acquisition* refers to the capture of information which supports knowledge-intensive choices. Acquisition may viewed as occurring in two distinct phases: initial acquisition and iterative refinement. *Initial acquisition* essentially involves capturing the knowledge which underlies choices 'from scratch'. *Refinement* involves incrementally modifying this knowledge. There are several reasons why a model of choice may require repairs (see Zeleny 1982), including errors in initial acquisition, changes in the attitudes of experts over time, and changes in the decision-making situation such as the introduction of new alternatives or objectives.

[5] Formally speaking, of course, deterministic computers do not admit *any* notion of arbitrariness. As used here, the term refers to the absence of any rationale underlying an ordering, in the same spirit as software manuals which warn of 'unpredictable results' for inputs which deviate from expectations.

Implicit models of choice suffice for intelligent system domains in which the process of choosing is not knowledge-intensive and hence is not considered to be an important part of the reasoning process *per se*. In such cases, there is no need to reason about choices, to explain the basis for choices, or to acquire new information which underlies choices.

But the need to choose among competing alternatives frequently arises in reasoning. Intelligent agents face explicit, knowledge-intensive choices in all sorts of settings, from the mundane and unimportant (choosing an activity on a Saturday evening, choosing an entree at dinner, choosing a shirt, choosing a seat in the living room) to the highly technical and critical (choosing a procedure for recovering from a fault in a nuclear power plant, choosing a therapy for treating a cancer patient, choosing the site for a new factory, choosing a route for trucking hazardous chemicals).

In domains in which choices are central, knowledge-intensive elements of the task that the program is designed to perform, explicit models of choice are appropriate. In particular, explicit models are employed to compute 'intelligent' choices, to provide a basis for automatically justifying choices, and to provide for systematically capturing and recapturing the knowledge which underlies choices. The purpose of the models, these supporting capabilities, and the relationship between them may be described differently depending upon whether one employs the model of choice to support the prescriptive or descriptive view of decision making.[6]

Taking the *prescriptive* view, we employ an explicit model to *tell us* how to choose in particular situations based on the information we provide about choosing for a general class of problems. This view requires that the model implement some rational theory of choice. The essential idea is that instead of encoding choices directly (i.e., implicitly), we encode the factors which underlie choices and rely on a model to combine these factors to arrive at a 'correct' choice. While the prescriptive perspective implies that the model's choices need not always agree with users' intuitions, it is not true that users should be expected to blindly accept them. Indeed, users will be inclined to believe the model's prescriptions only if convincing justifications for them can be generated. In cases where these justifications fail to convince the user, he will want to iteratively modify portions of the model (through an acquisition program) until the justifications for prescriptions seem more convincing.

In contrast, the *descriptive* view portrays the model of choice as a description of how *we* might choose. Under this view, an explicit model of choice serves as a device for predicting choices (Green & Srinivasan 1978) or for describing how choices are made (through an explanation facility). An explanation facility may also be used to verify that this description (model) is correct. The acquisition facilities provide the means to repair this description so that meaningful explanations and predictions may be generated.

In summary, many intelligent systems will need to employ explicit models of choice -- as either top-level or component structures -- because choosing among alternatives is a central element of reasoning in many domains. Such models are necessary to support the sound formulation of choices, the generation of justifications for these choices, and the modification of domain-specific knowledge which underlies these choices. We ground this general statement in specifics in the next section by examining the limitations of an intelligent system which employs an implicit model of choice where an explicit model would have been more appropriate.

2. A motivating application: intelligent process control

The proposed work was first motivated by the need for more effective *intelligent process control systems*, that is, systems which aid experts in (or completely automate) the management of complex physical systems such as nuclear power plants and large computer complexes. These systems are distinguished from more traditional control systems (e.g., Stephanopoulous 1984, Ray 1981) by

[6] For a thorough discussion see (Keen & Scott Morton 1978).

their employment of heuristic methods which mimic the reasoning of plant experts in addition to (e.g., Astrom 1986, DeJong 1983) or instead of (e.g., Chester 1984, Ennis et al. 1986) mathematical models of plant behavior and rigid control sequences for plant operation.

Work in this area is abundant, with applications in domains such as manufacturing (Wright et al. 1982), space systems (Scarl 1985), chemical processing (Chester 1984), nuclear power generation (Nelson 1982), computer operations management (Ennis et al. 1986), and several others. In fact, there has been sufficient interest in such applications to motivate the construction of special-purpose shells for intelligent control (e.g., PICON (Moore 1984), YES/L1 (Cruise et al. 1987)). In addition, there has been significant activity in the development of general representations for qualitative reasoning about physical systems (e.g., de Kleer & Brown 1984, Forbus 1984) which might prove useful in intelligent control applications.[7]

The need to make careful choices between competing alternatives frequently arises in the domains of intelligent control. For example, in performing diagnostic tasks, experts must carefully select among potential tests that might be initiated to ascertain the state of target system components so as to balance testing costs, the value of information yielded by tests, disruption to the target system and its environment, the safety of plant employees and of neighboring residents, and several other factors. In repairing physical systems (usually following a diagnosis), experts must often choose between numerous potential options ranging from temporary 'fixes' to the replacement of faulty components with new ones, guided by similar objectives.

One well-known example of an intelligent control system which encompasses such choices is YES/MVS (Ennis et al. 86), a forward-chaining rule-based system (implemented in OPS5 (Forgy 1981) and LISP/VM (1984)) which is designed to assist computer operators in the management of large industrial computer installations. YES/MVS is comprised of several 'domain specialists' which perform distinct tasks such as routine operations (e.g., swapping buffers, startup, shutdown), diagnosis and recovery from hardware and software failures, and job scheduling.

JESQ is a YES/MVS specialist which continually monitors and actively manages (MVS/JES3) operating system queue space, and exemplifies the complexity of making effective operational choices in intelligent control domains. JESQ served as the vehicle system for the author's master's thesis (Klein 1985), and its limitations with regard to effectively choosing among competing operational actions provided the initial motivation for the current enterprise. As such, we expose these limitations -- which characterize rule-based expert systems in general (Cromarty 1985, Sauers & Walsh 1983) -- in the remainder of this section.

A disclaimer: It is important to note that the JESQ (YES/MVS) project represented an investigation of research issues in realtime, active expert systems, *not* of representations for making knowledge-based choices. At the time of its development (beginning in 1982), YES/MVS was among the first realtime expert systems which exerted direct (closed-loop) control over its environment, and thus, we consciously focussed on issues concerning realtime reasoning and control.[8] Our strategy for making choices (and for addressing other requirements not related to realtime control, *per se*) was to use standard techniques (with little or no innovation) so as not to deviate from our research focus. The point of this disclaimer is that (i) JESQ is 'typical' in its approach to making choices in rule-based systems, (ii) this approach gives rise to several important problems for intelligent control and other knowledge-based systems, and (iii) the identification of these problems is not meant to discredit JESQ (or YES/MVS) in that work toward their solution was intentionally avoided in order to concentrate on other issues.

---

7 See (Klein 1986) for a detailed review and (Bobrow 1985) for a representative collection of papers.

8 For details regarding this investigation see (Klein & Milliken 1984, Ennis et al. 1984a,b,c, Milliken 1984, Milliken et al. 1985, Cruise et al. 1986a,b,c, Ennis et al. 1986, Klein et al. 1986, Chou et al. 1986, Cruise et al. 1987). For a general discussion of the requirements of active expert systems that was inspired by the project see (Klein & Finin 1987).

The following section provides the background knowledge needed to appreciate JESQ's task. This description also serves to impart the domain knowledge upon which most examples in the proposal are based, so we ask the reader to bear with us. Next, we provide a detailed description of JESQ's architecture and its limitations, which serve, in part, as practical motivation for the proposed thesis. A more detailed description appears in (Klein 1985).

2.1 JESQ's domain

Job Entry Subsystem (JES) queue space is a common resource (disk storage) in IBM system environments for the staging of computer jobs before, during and after execution. Jobs are normally deleted from the queue space once output has been completed to a printer, a transmission line, or other output medium. JES queue space is also used by JES itself as a scratch area for executing its functions. In addition, JES maintains batch job output for online viewing (via IBM's Time Sharing Option (TSO) software) in the JES queue space area.

Operations management is concerned with monitoring the amount of available queue space because its depletion requires restarting the system, potentially inconveniencing all system users for a substantial period of time. Of course, the problem could be eliminated from time to time by employing the 'brute force' strategy of allocating more and more disk storage to JES as needed (although this allocation is fixed at system startup time). But this tradeoff of effective space management (labor already paid for) for additional physical storage (capital) is naturally frowned upon by management, representing an expensive and temporary 'fix' in the absence of identifiable increases in system workload.

The operator may take several protective and corrective actions when queue space begins to diminish, and these may be described in terms of three general goals:

- *Protect remaining queue space*: The operator must protect the space that remains when dangerously low (e.g., 5%). For example, the operator may vary the main processor offline, blocking the initiation of additional jobs which could generate output on the queue.
- *Free queue space*: The operator can manipulate various devices and operating system parameters to free queue space. For example, the operator may run **DJ** (for **D**ump **J**ob) to copy large jobs from the queue to tape, and then reinstate them for printing once the queue space situation has improved. Alternatively, the operator may change parameter settings on printers to allow jobs with special characteristics (e.g., special paper or security requirements) to print. The operator may also change the maximum line count limits on printers set to favor small jobs in cases where large jobs are waiting and small jobs will soon all be printed. The operator can additionally reroute large jobs destined for slow printers to faster printers with a relatively light load.
- *Diagnose and eliminate the cause(s) of queue space depletion*: In some cases, there exists a direct cause-effect relationship between the actions of an environmental agent (e.g., user, operator, device) and a queue space problem. For example, a printer might not be operational, or a link to another system might be down. In such cases, the operator must correct the problem as well as restore the queue to an acceptable state in a reasonable amount of time.

Significant judgement is required of the operator in choosing among competing actions. For example, output stored for online (TSO) viewing can be purged from the queue by using DJ, by requesting action from the user himself, by printing the job, or even by deleting the job. In general, choices between competing actions are based on a set of underlying decision criteria which includes:

- *anticipated impact* on queue space resulting from the successful execution of the action;

- *operator convenience*, including the amount of time spent by the operator in executing an operational heuristic and the amount of 'work' involved;[9]
- *material cost* of the action in excess of originally scheduled processing;
- *user satisfaction*, including considerations of user turnaround time, the additional time expended by the user himself in accomplishing his processing goals,[10] and the difference in the quality of his output from that requested; and
- the *speed* with which actions may be executed.

As no event in the computer operations environment is certain, it is the case that probabilistic factors also come into play. For example, the 'recent success' of particular actions might factor into the decision when some facility is not correctly operating, as in the case of a device which seems to 'ignore' commands issued to it. Given that some devices exhibit this behavior more than once, we might additionally consider that the 'track record' of actions over the cumulative history of their execution be included as a factor which underlies operational decisions. But practically speaking, we should more or less ignore such probabilistic concerns in this domain, for if they become predominant considerations, the devices which give rise to them should be replaced. In any event, it would be very difficult (and certainly not worth the effort) to develop probability distribution functions to describe such behavior. As any model represents an abstraction of reality, in this domain it is appropriate for the process of choosing to reflect the assumption that the outcomes of actions occur with certainty.

2.2 Organization of JESQ's knowledge base

As in most rule-based systems, the essential unit of knowledge in JESQ is the rule. Our goal was essentially to map each operational heuristic recorded in the installation's run book[11] directly into a rule, and to encode a standard set of rules for performing supporting tasks such as querying the status of the target system. In this way, the benefits of modularity and mutual independence of heuristics often associated with the rule-based paradigm would be realized, allowing the installation to add, modify, and delete heuristics with ease as the installation evolved. As will be described, most of the JESQ's limitations are due to its inability to select the *best* heuristics at any point in time.

Rules in JESQ are grouped along two orthogonal dimensions: by function (e.g., query submission, information collection) and by problem severity (as a function of space left on the queue), as described in the following sections.

2.2.1 Rule grouping by function

JESQ's rules are grouped in functional classes. Each functional class is associated with a priority which determines which rule will be invoked when rules from more than one class are concurrently satisfied in a given iteration of the recognize/act cycle.[12]

JESQ's rule groups include:

- *System Initialization and Control*: This group contains rules that create the abstract internal model of the target system environment, enable and disable groups of rules as a function of the

---

9 Time and work are not equivalent in this context. For example, most operators would rather spend 5 minutes submitting commands through their consoles than spend the same 5 minutes moving heavy boxes of paper.

10 Resubmitting jobs deleted by operators and talking to operators over the phone are two examples of actions which consume users' time.

11 A *run book* is a list of procedures supplied to operators which describes the appropriate courses of action for dealing with anticipated problems and routine requirements.

12 We augmented OPS5 conflict resolution with a priority mechanism. The set of satisfied rules is first reduced to contain only those of the same priority, and the resulting set is resolved on the basis of recency of information and specificity of antecedent conditions (i.e. OPS5 conflict resolution).

severity of the queue space problem at hand, and suppress certain actions when specified by the operator.

- *Periodic Query Submission and Timeout Handling*: This group controls the periodic querying of target system resource states. Query intervals are based on estimates of the reliability over time of the information being captured. Rules are also included to resubmit queries that have been lost in transmission (i.e., timed out).
- *Information Collection and Data Reduction/Expansion*: This group includes rules that collect target system messages and update JESQ's internal model accordingly. Portions of this abstract model appear in the antecedents of the Knowledge-Based Action rules which take space management actions. Some rules in this group map a single response working memory element (wme) into a single internal model wme. Other rules perform data reduction, manipulating multiple response wmes to produce a single summary wme that is referenced by the Knowledge-Based Action rules. Still other rules perform data expansion, supplying attributes with values that are only implied by target system responses.
- *Miscellaneous Cleanup and Response Collision Collection*: This rule group deletes target system responses and expert system-generated goals from working memory. Rules in this group also delete asynchronously arriving responses to duplicate queries that have been delayed by failing or sluggish target system resources.
- *Knowledge-Based Action*: The above described groups exist to support the Knowledge-Based Action rules which encode queue space management policy. Rules are included to protect the remaining queue space, to set up for space-freeing actions, to reset target system parameters when space returns to acceptable levels, to free queue space when a problem exists, and to alert the operator to potential problems that cannot be further diagnosed without additional information. These rules are further decomposed into three subgroups of varying priority: *low-, medium-,* and *high-priority-knowledge-based-actions.*

Thus priorities are used for two purposes in JESQ: (i) to proceduralize the execution of rule groups and (ii) to indicate the relative desirability of plans encoded by Knowledge-based Action rules. The limitations of interest in this section concern the latter usage.

2.2.2 Rule grouping by problem severity

Groups of rules are enabled/disabled dynamically during expert system execution according to the severity of the queue space problem at hand. For example, a drastic action such as varying the main processor offline is appropriate when only 3% of the queue space remains, but not when 10% remains. To implement this knowledge, thresholds of space left are mapped to five symbolic *processing modes* (NORMAL, WATCH, POKE, SOLVE, and PANIC), each associated with range of space left on the JES queue. Some actions are limited to a single processing mode (e.g., varying the main processor offline). Other actions span multiple processing modes (e.g., raising the line limit on a printer).

2.3 Critique of JESQ

Having described JESQ's domain and its architecture, we can now proceed to expose the limitations of that architecture with respect to its domain-specific requirements.

JESQ surely takes reasonable actions; the system ran successfully at IBM's Watson Research Center for most of a year and received a favorable response from operations staff. But we have no justification for believing that JESQ takes the *best* actions at any point in time because JESQ contains no explicit model for making choices. The relative desirability of knowledge-based actions is represented by the three priority levels (low, medium, high), and the assignment of these priorities to individual heuristics takes place outside the system. Because selecting among competing

heuristics is a complex and knowledge-intensive task, we have no reason to believe that assigning priorities 'by the seat of the pants' produces optimal results.[13]

Another problem with JESQ's selection scheme is that it lacks robustness. The overall behavior of JESQ is extremely sensitive to the assignment of priorities. In effect, the complex set of considerations which underlie the selection of priorities (see section 2.1) have been bundled into a single symbol. Given that priorities are assigned with some degree of arbitrariness, JESQ's behavior may be described as somewhat arbitrary.

Another limitation of JESQ concerns it transparency. While JESQ provides (canned) explanations regarding how recommended actions achieve the goals of queue space management, it offers no justification as to why particular actions are preferred to others. Again, this is because there is no explicit model of choice in JESQ. Since the factors underlying priority assignment are not represented in the system, the best explanation that JESQ might generate would be to merely display the priority of the chosen heuristic or to compare its priority with the priority of another. Because JESQ cannot justify its choices, operations managers have no basis for deciding if JESQ's operation correctly reflects the goals of the installation or for identifying how its knowledge base might be enhanced.

JESQ's most objectionable flaws concern the difficulty involved in integrating new heuristics with existing ones, again due to the lack of an explicit model of choice. Since the considerations underlying the selection of competing heuristics are nowhere represented in the system, changes to the knowledge base must be addressed as a programming task. It is up to the knowledge engineer to hack the rules such that the desired behavior is achieved, if in fact that behavior can be identified. In order to intelligently manipulate the priority of a rule in JESQ, it is required that the knowledge engineer understand the basis for the priorities of all existing rules in the knowledge base and that he be able to envision all potential conflict sets of interest. Formulating priorities for new heuristics is especially difficult, in that the considerations underlying priority selection may be forgotten by the knowledge engineer over time so that priorities are not assigned according to any consistent scheme. If multiple knowledge engineers maintain the system, this will almost surely be the case. Thus, while it is certainly easy to augment or change the rules in JESQ's knowledge base, it is almost impossible to ensure that rules will be invoked at the proper points of execution.

The ability to modify the knowledge base is especially important in JESQ's domain, where the environment -- and hence, the knowledge concerning its control -- is subject to frequent change. Typical changes in the real world that are reflected in the way the installation is managed include, for example, changes in the installation's abstract goals (e.g., the introduction of new safety standards), changes in the relationship between those goals (e.g., increased cost consciousness, perhaps at the expense of quality of service), changes to the target system configuration which create new operational alternatives (e.g., the introduction of a new printer to the machine room), changes to the target system configuration which modify the characteristics of existing operational alternatives (e.g., the replacement of parts on existing printers), and others. Given that intelligent control systems like JESQ will contain hundreds (or in some cases thousands) of operational heuristics, the integration of new or modified heuristics cannot be pragmatically viewed as a programming-level task.

From an operational viewpoint, then, we need to be able to view JESQ as a *storehouse of transparent, evolving heuristics* for managing queue space which reflects the current goals of installation management at any given time. Our inability to do so is principally due to JESQ's lack of an explicit model of choice. JESQ thus provides a convincing case study which supports the hypothesis

[13] This is confirmed by experiments in which we presented several operators with the same description of an operational situation and a set of alternative actions. We asked the operators to rank those actions and failed to receive the same rankings from all the operators. Given that *some* ranking is optimal, it must be the case that some of the operators produced suboptimal rankings.

that at least some intelligent systems require an explicit model of choice. The inclusion of such a model would have provided a basis for organizing JFSQ's numerous operational heuristics and for justifying those heuristics to operations managers. But what sort of explicit model of choice might have best served these purposes? What properties render explicit models of choice useful for intelligent systems? These questions are taken up next.

3. Desiderata for explicit models of choice in intelligent systems

Explicit models of choice are necessary to support the computation, explanation, and refinement of complex choices in intelligent systems. In this section we enumerate some of the characteristics of models of choice which facilitate the support of these capabilities. To summarize the discussion that follows, these include:

- *competence:* The notion of *best* in a model of choice should be well defined. The model should be robust, meaning that the outcome of choosing does not rely on a single or just a few symbols which may not have been accurately captured. The model should be economical, in both its storage and processing requirements. The model should be general, with application beyond a single or only a few domains.
- *transparency:* The model should be comprised of objects which are meaningful to users in isolation and are combined in an intelligible way. The model should be composable so that the level of detail in explanations can be varied as users desire.
- *ease of construction and evolution:* There should be some systematic way to build the model. Moreover, the model should support graceful extension and modification, i.e., it should be possible to add or modify only those portions which must be molded to reflect reality. The model of choice itself should provide for the graceful extension of the set of objects from which it chooses.

We elaborate on these desiderata in the following sections.

3.1 Competence

First, the notion of *best* in a model of choice should be well defined. The behavior of the model must reflect some underlying theory and be based on some agreeable set of assumptions. If it is not, we have no basis for understanding why it produces seemingly correct results when in fact we intuitively agree with those results. More importantly, we have no basis for understanding if or why it fails when the results are counterintuitive. In the latter case, we are forced to 'hack' at the model until the desired behavior is achieved. Under the prescriptive view, we don't even know what this behavior should be, and hence, what sort of hacking we should do.

Second, the model of choice should be *robust*, meaning that the outcome of choosing does not solely depend on a small set of subjectively assigned values. Rather, we want such elusive quantities as choice-related preferences to be distributed over a larger set of symbols (each with finer-grained semantics) so that small errors in model inputs change the overall results accordingly, or ideally, cancel each other out.

Third, the model should be *economical*, in both its storage and processing requirements, since models of choice often play a central role in processing. For example, the conflict resolution algorithms of production systems are invoked on every iteration of the recognize/act cycle. Another example, heuristic evaluation functions are executed with each move in game-playing programs.

Finally, the model should be *general*, with application beyond a single or only a few domains. If we employ customized models for every intelligent system we build, we require knowledge engineers to develop an understanding of each and every model employed. In addition, we are forced to build

acquisition and explanation facilities for several models, which in turn force intelligent system users to think in terms of several models.

3.2 Transparency

The model must provide the basis for justifying choices. As transparency is a critical determinant of user acceptance for intelligent systems in general (Teach & Shortliffe 1981), it is especially important for justifying the choices computed by normative models, since those choices may, on the surface, seem counterintuitive. The justifications must themselves, however, be intuitively appealing in order to be convincing. Thus, the model must explicitly represent information which will convince users that a chosen alternative is the best one, i.e., they must contain the 'right stuff', and in isolation.

We would also like our justifications to focus upon the 'important' aspects of choice in any particular situation, while providing the user with the flexibility to probe further into any aspect of the choice which will better confirm or deny the model's results. Thus, we advocate models of choice which are composable so that choices may be justified in logical fragments.

Finally, it is desirable for the combining operations of a model choice to be intelligible so that users can comprehend how isolated elements of the model influence its overall behavior.

3.3 Ease of construction and evolution

There must be some systematic way to build the model. The structure of the model itself must suggest the type of information to be encoded within it.

Moreover, the model should support graceful extension and modification. As the knowledge underlying complex choices will have to be iteratively captured, it should be possible to modify only those portions which must be molded to reflect reality. Thus, we require that pieces of the model be composable so that subproblems of the overall choice problem may be addressed in isolation. In particular, we require the capability to (i) add new factors which underlie the choice to the model as they are identified, (ii) specify existing factors in more detail as required, and (iii) change the relationship between existing factors as required.

Finally, the model of choice must provide for the graceful extension of the set of objects from which it chooses. This is among the primary motivations for employing an explicit model of choice: The model should allow us to describe an arbitrary number of objects from which to choose, specifying only their characteristics in terms of the factors which underlie their selection. The model of choice itself thus serves as a mechanism for logically integrating new objects into the set to be chosen from. We should be able to describe the objects in isolation and let the model of choice do the rest.

4. Decision-theoretic models and the AMVM

The framework of decision analysis is concerned with choosing among alternatives in an uncertain environment.[14] The framework provides for the systematic treatment of the *utilities* of outcomes resulting from alternative courses of action and their associated *probabilities*. Keeney (1982) describes decision analysis as 'a formalization of common sense for decision problems which are too complex for informal use of common sense', and more technically as 'a philosophy, articulated by a set of logical axioms, and a methodology and collection of systematic procedures, based upon those axioms, for responsibly analyzing the complexities inherent in decision problems'. In this section we briefly review decision-analytic models, concentrating on the *additive multiattribute value*

[14] This entire section is expository in nature. Those familiar with decision theory should skip to the next section.

model (AMVM), which will be the focal model of the proposed research. The following presentation is abstract; a detailed sample AMVM is presented in chapter V.

Keeney and Raiffa (1976) classify decision problems along two dimensions. The first, attribute multiplicity, concerns the number of attributes that underlie the choice at hand. In this context, *attributes* are measurements (e.g., dollars spent) that characterize alternatives with respect to the *objectives* (e.g., minimize cost) which drive their selection. Some choices rest upon a single attribute, while others are best expressed with regard to multiple, often *mutually competitive* attributes (e.g., the classic 'quality vs. quantity' dilemma). In the latter case, the collection of attribute values for an alternative represents a sort of 'profile' for that alternative. The second dimension concerns the certainty with which the potential outcomes of a decision are known. This view of choices gives rise to four choice types of potential interest: single-attribute choices under certainty, multiattribute choices under certainty, single-attribute choices under uncertainty, and multiattribute choices under uncertainty.[15]

Clearly, the fourth type of choice is the most general and interesting. But we wish to start with a simpler model that will provide a testbed for our ideas about explanation and refinement of decision-theoretic models, with an eye toward generalization to more complex and general models. The proposed work will be concerned only with the multiattribute case under certainty. In the context of this model, the problem of choosing is sometimes called the *multiattribute value problem* because the choice focuses only on the *values* of the outcomes of actions, assuming those outcomes are known with certainty. This restriction is, of course, suitable for modelling decisions in some domains (e.g., managing queue space, where the certainty of outcomes of actions plays a role in decision-making, but not a central one) and less suitable in others (e.g., some domains of therapy planning in medicine, where the lack of certainty of treatment outcomes is at center stage in choosing among alternative treatments).

The multiattribute value problem may be stated as follows.[16] Let a designate a feasible alternative and denote the set of all such alternatives by A. To each act a in A we will associate n indices of value: $X_1(a), \ldots, X_n(a)$. We can think of the n evaluators[17] $X_1, \ldots, X_n$ as mapping any given a in A into a point in an n-dimensional *consequence space*.

Roughly, the decision maker's (or intelligent system's) problem is to choose a in A so that he will be happiest with the payoff $X_1(a), \ldots, X_n(a)$. Thus we need a mechanism that combines $X_1(a), \ldots, X_n(a)$ into a scalar index of preferability or value. Alternatively stated, it is adequate to specify a scalar-valued function v with the property that

$v(X_1(a), \ldots, X_n(a)) \geq v(X_1(b), \ldots, X_n(b))$ iff a is *preferred or indifferent to* b.

We refer to the function v as the *value function*.[18] Given v, the decision maker's (intelligent system's) problem is to choose a in A such that v is maximized. The value function v serves to compare various levels of the different attributes indirectly. The formal correctness of the choices yielded by such models lies in the axioms and theorems that constrain the behavior of v, which are not reviewed here.[19]

---

15 Some readers will be disturbed by the omission of purely probabilistic choices, but these have indirectly been included: It was shown by Ramsey (1926) how one could build up the theory of probability by starting from the principle of maximizing expected utilities. The argument is paraphrased in (Good 1983).

16 Part of this description is adapted from (Keeney & Raiffa 1976), pp. 67-68.

17 Following (Keeney & Raiffa 1976), we avoid distinguishing between an *attribute* X and an *evaluator* X for this attribute, relying on the context of the discussion to resolve any ambiguity. It is generally clearer not to draw distinctions between these two concepts.

18 There is some confusion regarding this term in the literature. The same construct is also referred to as a *preference function*, *worth function* and *utility function*. Some authors distinguish between *value function* and *utility function* as corresponding to the cases of certainty and uncertainty (of outcomes) respectively.

19 See (Keeney & Raiffa 1976) for a thorough exposition.

In this work, we further limit our interest to problems where the attributes $X_1, \ldots, X_n$ are *mutually preferentially independent*. Informally, this means that the tradeoffs between every pair of attributes, keeping the levels of other attributes fixed, do not depend on the particular values of these fixed levels.[20] This restriction permits the use of the following special form for v, called the *additive form*:

$$v(a) = v(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i v_i(x_i)$$

where:

1. Each a in A is represented by a vector of attribute values $(x_1, \ldots, x_n)$;
2. v_i is the *component value function* for attribute i, with $v_i(\text{worst } x_i) = 0$, $v_i(\text{best } x_i) = 1$, and $0 \leq v_i(x_i) \leq 1$ for all x_i;
3. w_i is a *scaling constant* or *weight* for attribute i, $0 < w_i \leq 1$ and $\sum_{i=1}^{n} w_i = 1$.

Informally, the weights indicate the relative importance of each attribute as it changes from its best to its worst value. The component value functions express the relative desirability of various levels of their respective attributes.

While the additive form rests on strong assumptions -- which must be verified with decision-makers -- its use is standard in practice. According to Zeleny (1982), 'the additive and multiplicative utility functions are both simple and robust approximations, and they are the only practical options for cases with more than four attributes'. Dawes (1979) presents evidence that linear models are 'superior to clinical intuition' in predictive settings. Keeney (1986) maintains that 'when the objective functions are complex, meaning they involve more than additive or multiplicative components of single-attribute objective functions, it is often the case that the original objectives were not wisely selected'.[21] Thus, while a restricted version of the general decision-analytic paradigm, the AMVM is sufficiently general to have broad potential application, spanning decisions across several domains.

5. Assessing decision-theoretic models for intelligent systems

We evaluate decision-theoretic models with respect to the general desiderata of section 3 again organizing the analysis in terms of competence, transparency, and ease of construction and evolution. The discussion is intended as general, but we distinguish the AMVM where appropriate.

5.1 Competence

We can present a strong case regarding the competence of decision-analytic models. First, the paradigm of decision analysis provides a formal foundation for making complex decisions. As Keeney (1986) notes, 'The relative strength of decision analysis is that it has a sound foundation provided by axioms stated in von Neumann and Morgenstern (1947), Savage (1954), and Pratt et al. (1964), and sound procedures to implement this logic'. The paradigm thus provides principled, domain-independent machinery for making choices. This means that we can pinpoint the assumptions that must hold in a domain in order for a particular model to produce correct results, and that we have a basis for understanding why the model works.

---

[20] See (Keeney & Raiffa 1976) for a more formal discussion. Other functional forms following various assumptions can be found in (Fishburn 1970), (Meyer 1970), (Bell 1979), (Tamura & Nakamura 1978), and (Farquhar & Fishburn 1981).

[21] See (Keeney 1981) for typology of ill-selected objectives.

Decision-analytic models also provide the robustness we seek in that they generate results based on a distributed set of symbols which represent the values and probabilities of the potential outcomes of the alternatives under consideration. This is to be contrasted with choice-making schemes which rely on a single or just a few symbols to which the behavior of the reasoning paradigms encompassing them may be highly sensitive.

Third, decision-analytic models are relatively economical, in both their storage and processing requirements. The models are cast as relatively simple mathematical expressions which can be quickly executed and require very little storage. The parameters of these expressions are likewise not particularly memory-intensive. Decision-theoretic models are to be contrasted in this sense with formalisms such as meta-rules (Davis 1976, 1980) which rely on processor- and memory-intensive operations (pattern matching) for their execution.

Finally, decision-analytic models implement a general theory of choice with potential application in any domain where the alternatives, outcomes, and the values and likelihoods of outcomes can be identified. Reported applications of decision analysis span a diverse set of domains and decisions, including the examination of corporate policy (Keeney 1975), evaluation of capital investment options (Magee 1964), budget allocation (Keefer & Kirkwood 1978), credit application evaluation (Stillwell et al. 1980), medical decision making (Krischer 1980), hurricane seeding (Howard et al. 1972), metropolitan airport development (de Neufville & Keeney 1972), fire protection (North et al. 1975), school busing (Edwards 1980), oil tanker standards (von Winterfeldt 1982, Ulvila & Snider 1980), nuclear waste management (Lathrop & Watson 1982), commercialization of solar photvoltaic systems (Boyd et al. 1982), siting of energy facilities (Keeney 1980, Sarin 1980), and many others.

Let us defend some anticipated counterarguments regarding the competence of decision-theoretic models. First, some might conjecture that since decision-theoretic models are cast in mathematics, they require 'too much precision' for use in practical intelligent systems. After all, one of our goals in developing such systems is to allow fuzzy judgements with incomplete information. But this objection is entirely without merit, because the mathematics provides only the *machinery* for computing choices; the *information encoded* in the mathematical model for any particular domain-specific choice may be as vague or as detailed as the domain warrants. In addition, recent research in decision theory provides methods which explicitly account for incomplete information in a systematic fashion (Weber 1985, Weber 1987).

Second, a common criticism is that decision-theoretic models are 'too restrictive' or 'idealized', that 'too many assumptions' are required. Certainly, classes of decision-theoretic models rest on associated classes of assumptions; in fact, *all* models (of choice and otherwise) rest on some (hopefully well-specified) set of assumptions. But it is because we can identify *what those assumptions are* (i.e., the axioms of decision theory and the particular assumptions underlying particular forms of the model) that, in part, makes decision theory an attractive model of choice for intelligent systems. As an abstraction of the complex decision making process, decision theory is surely limited, but in an accountable way. This is to be contrasted with implicit models of choice and with *ad hoc* explicit models, the limitations of which remain obscure.

A related objection concerns the validity of even the most basic assumptions of decision theory (i.e., the axioms); the claim is made that people do, in fact, violate them in choosing between competing alternatives. But Keeney (1982) argues that many decision makers *prefer* to act in accord with the axioms. That decision makers seriously violate those axioms in choosing alternatives without the benefit of decision analysis is an argument *in support of* decision analysis. In Keeney's words, (1982) 'The purpose of prescriptive decision analysis is to provide insight about which alternative should be chosen to be consistent with the information about the problem and the values of decision makers'.

Finally, it is sometimes argued that a decision-theoretic model will always leave out some important factor which underlies a choice. This is indeed a valid concern, but one which applies to *any* model of choice. In addition, decision analysis provides for calculating the value of additional information (LaValle 1968, Merkhofer 1977). With less formal models, we generally have no basis for assessing the value of new information which may be expensive to obtain. Thus, concerns regarding incomplete information give rise to arguments for, rather than against, the competence of decision-theoretic models.

In summary, decision-theoretic models fare well with regard to our competence-related desiderata; they are grounded in a well-formed theory, are relatively robust representations of preferences, are relatively economical, and are broadly applicable.

5.2 Transparency

One frequently espoused argument against the use of decision-analytic models in intelligent systems is that they are not particularly well-suited for automating explanation; They are 'too quantitative' or 'too complex' for exposition, as the story goes. Consider the following counterarguments.

First, decision-theoretic models explicitly represent the component values that underlie decisions. Again quoting Keeney (1986), 'Values are the basis for any interest in any decision problem. Why is it worth the effort to carefully choose an alternative rather than simply let occur what will? The answer is that some concerned party is interested in the possible consequences that might occur. The desire to avoid unpleasant consequences and to achieve desirable ones, especially when the differences in the relative desirability of the consequences is significant, is the motivation for interest in any decision problem. The relative desirability of the possible consequences in decision problems is based on values.' As values are explicitly represented in decision-theoretic models, they are available to display for users in an explanation or justification. As will be discussed, the interesting questions in automating justifications for choices involve intelligently *pruning* and naturally *organizing* the set of values that are displayed for users, rather than *generating enough* information to display.

In addition, decision analysis provides for decomposing choice problems into subproblems that can be separately analyzed and integrated according to the logic of the axioms. As will be described, the objectives that underlie decisions may be naturally cast in a hierarchical arrangement (called the *objectives hierarchy*), providing a basis for varying the level of detail in explanations according to the properties of the choice at hand, and allowing the user to interactively control the level of detail during the course of an explanation.

In addition, simple, restricted models (such as the AMVM) provide an intuitively appealing framework for thinking about choices. In the AMVM there exists a natural correspondence between the operands of the model's component products (weights and values) and ideas involved in choosing: weights correspond to the *importance* of an attribute,[22] and values express how *(un)desirable* particular levels of attributes for a given alternative might be.[23] Second, the products themselves may naturally be thought of as *contributions* to the overall evaluation of an alternative. Quite simply, then, the greater the value (desirability) and the weight (importance) of an attribute, the greater the attribute's contribution. The greater the contribution of each attribute, the better the alternative comes out in the evaluation.

Thus, we claim that decision-theoretic models are in fact transparent in the sense of section 3.2. Our mission is to exploit this transparency in constructing an effective explanation facility.

---

22 Technically, the relative importance of an attribute as it changes from its best to its worst value.

23 With respect to a predefined range of possible levels appropriate to the problem at hand.

5.3 Ease of construction and evolution

The initial construction of virtually all models of choice remains more an art than an engineering discipline. While problem structuring is an open problem (although some work has been done, e.g., Jungermann 1980, von Winterfeldt 1980), decision theory at least provides a systematic set of procedures for capturing the parameters of decision-theoretic models (e.g., see (Fishburn 1967), (Huber 1974), (Keeney & Raiffa 1976), (Farquhar 1984)), some of which are sufficiently algorithmic for implementation in computer programs (e.g., (Keeney & Sicherman 1976), (Nair & Sicherman 1979), (Novick et al. 1980), (Schlaifer 1971), (Seo et al. 1978), (Klein et al. 1982), (Weber 1985), and (von Nitzsch & Weber 1986)). Such programs are possible because decision analysis provides a *theory* of choice; since the semantics of model parameters are well-defined, their capture is a well-formed task. This is to be contrasted with *computational formalisms* in which such theories might be encoded (e.g., meta-rules), but which themselves say nothing about the nature or elements of choice knowledge.

On a less positive note, it is generally agreed that capturing the parameters of decision-theoretic models is an extremely effort-intensive task (e.g., see Zeleny 1982, Keeney 1986), and this, on the surface, might lead us to frown upon the employment of the models in intelligent systems. But this would ignore the very important operational differences between the standard paradigm of decision analysis and that of intelligent systems. The capture of preferences is usually an expensive process because decision analysis is most frequently employed in the context of critical one-shot decisions (e.g., see (Keeney 1982, von Winterfeldt & Edwards 1986)). For such decisions, where lives or great sums of money may be at stake, great effort is required in the acquisition phase because the resulting model must be viewed with a high level of confidence before it is ever used (usually only once). In the context of intelligent systems, however, we may settle for a less reliable initial model because this model will be repetitively used over time. Provided that we can offer users adequate facilities for incrementally modifying models as they observe system behavior, they need not 'get it right' the first time around. Thus, we might be able to claim that decision-theoretic models support ease of construction if we can argue that they potentially support ease of evolution.

There are several reasons to believe that the models support evolution. First, decision-theoretic models explicitly represent the component values that underlie decisions. As values are the essential ingredients of choices and are isolated in the models, there exists the potential to modify them in isolation.

In addition, the decomposition of choice problems into subproblems that can be separately analyzed and integrated according to framework of decision theory provides for focussing on subproblems that are deemed to be suspect.

Again we mention that some models, particularly the AMVM, provide an intuitively appealing framework for thinking about choices, thus establishing a natural correspondence between the structures that must be captured from users (e.g., weights, values) and conceptual entities in the user's mind (importance, desirability). This natural correspondence between conception and computation is precisely what makes automated refinement possible. Our job, of course, is to find the best way to help users focus on the appropriate portions of the model and to make it convenient to repair those portions when necessary.

Thus, we argue that decision-theoretic models, particularly the AMVM, support ease of evolution, and hence, reduce the demands of construction.

Finally, decision-theoretic models support the incremental modification of the set of alternatives for selection. All that is required to add new alternatives is that the values of attributes be specified, and that these values fall in the prespecified ranges assumed in formulating the decision-theoretic model.

Chapter III: Research Goals and Issues

Having motivated the problem, we explicitly state the ultimate research goals associated with the current enterprise (section 1), and then circumscribe a modest but coherent portion of it which will constitute the thesis (section 2).

1. Ultimate research goals

The general goal of this work is to provide the basis for employing formal models of choice in intelligent systems, and this encompasses two challenging tasks. The first is to develop a comprehensive, integrated system for:

1. helping users to structure decision problems,
2. acquiring initial decision-theoretic models from users,
3. justifying choices based on these models during system operation, and
4. helping users to incrementally modify these models on an ongoing basis as errors are uncovered and as preferences change.

The ideal system would access general storehouses of knowledge to help users identify the objectives associated with arbitrary problems. The system would select a specific form of utility function for the problem at hand, and aid in the specification of this utility function. The system would produce concise, convincing explanations for its choices and would make it convenient for users to focus on precisely those portions of decision-theoretic models which need to be repaired in order to reflect reality and to maintain internal consistency.

The second major task involves the development of a theory which describes how decision-theoretic models should be integrated with other knowledge structures. The theory would provide specific guidelines regarding:

1. how decision-theoretic models should be integrated with more encompassing decision-making frameworks (e.g., rule-based systems, theorem provers, reasoning paradigms based on 'deep models'),
2. how explanation facilities for decision-theoretic models should be integrated with explanation facilities for the other representations, and
3. how acquisition facilities for decision-theoretic models should be integrated with acquisition facilities for the other representations.

Such a theory would be a component of a more encompassing theory of knowledge representation which provides well-defined guidelines for matching reasoning techniques to intelligent tasks and specifies the associated implications for integrating their supporting facilities (e.g., explanation and acquisition).

The first task involves building a domain-independent module for capturing, using, and maintaining knowledge about choices. The second task involves developing a domain-independent set of specifications for using this module in concert with others.

2. Research goals of the thesis

Addressing these long-term research goals represents a few lifetimes of work. We will therefore limit our attention to the following subgoals in the thesis.

First, we propose the development of a domain-independent collection of mechanisms for explaining choices based on AMVM's. Second, we propose the development of a domain-independent collection of mechanisms for helping users to refine existing AMVM's. Together, the explanation and refinement facilities will comprise UTIL.

Third, we propose to briefly address the integration of decision-theoretic models into some popular intelligent system architectures and to discuss the integration of UTIL with their respective facilities for explanation and refinement.

We proceed as follows in elaborating upon these objectives. Section 2.1 discusses the purpose of and the design goals for UTIL. Sections 2.2 and 2.3 describe research issues in explaining and refining decision-theoretic choices, respectively.[24] Section 2.4 discusses issues of UTIL's integration with analogous facilities in architectures where decision-theoretic models coexist with other knowledge structures.

2.1 UTIL and the explanation and refinement of AMVM's

We propose to develop a set of mechanisms, collectively referred to as UTIL, which provide a domain-independent, integrated framework for explaining and refining AMVM-based choices.

In chapter II we described two distinct phases of knowledge acquisition: initial acquisition and refinement. We have in mind the following strategy for distributing the user's effort over these two phases in intelligent systems. The approach to initial acquisition should be as simple as possible, since UTIL will presumably provide for the convenient repair of erroneous portions of the model as the intelligent system which includes it is used. Of special interest are initial capture methods which do not force the user to 'think hard' about tradeoffs between objectives, but rather, require only holistic judgements over small sets of representative alternatives from which the 'part worths' of levels of attributes underlying a decision are inferred (e.g., see (Green & Srinivasan 1978) for a review). Also of extreme potential value in this regard are initial acquisition methods which accommodate incomplete information in a structured fashion (e.g., Weber 1985, Weber 1987). The idea behind the proposed strategy is to capture only a rough approximation of the utility function in initial acquisition, initiating subsequent refinements as necessary during the life of the intelligent system, when the user is already thinking about why a particular choice is erroneous. Thus, UTIL represents an integrated approach to initial acquisition, explanation, and refinement that treats choice models as transparent, evolving representations of users' preferences.

UTIL should support both the prescriptive and descriptive frameworks for making choices. From the prescriptive point of view, UTIL's explanation component will serve as a window into how choices are made. In cases where users fail to find UTIL's justifications convincing, they will iteratively invoke UTIL's refinement facilities to 'repair' portions of the model until convincing justifications are generated.

From the descriptive viewpoint, users may employ a holistic method to capture initial preferences, relying on UTIL's refinement facilities to repair the underlying basis for what *they* see as the 'best' choice. In this scenario, UTIL's explanation facility it at center stage; the user already knows what the best choices are, and the underlying model serves as a basis for arguing why these are best to other users.

UTIL is intended as a set of modules which may be invoked in the context of any architecture which supports complex, knowledge-based choices. We approach the construction of UTIL assuming that an architecture-dependent supervisor invokes UTIL modules and other modules which

[24] While we address them separately for purposes of exposition, it should have by now been communicated that explanation and refinement are tightly integrated tasks.

support the explanation and refinement of other structures with which the AMVM may coexist (e.g., production rules, deep models of mechanism).

In summary, UTIL is intended as a general, domain-independent set of mechanisms which supports the explanation and refinement of AMVM-based choices. We view UTIL as a set of modules which:

- should be compatible with existing automated methods for initial acquisition,
- should support both the prescriptive and descriptive views of decision making, and
- should allow for integration with other facilities that support structures commonly used in intelligent systems.

UTIL raises several research issues regarding the explanation and refinement of AMVM's: What makes a justification 'convincing'? How can we avoid the complete reformulation of an AMVM when portions of it become outdated? How can explanation and refinement facilities for AMVM's be integrated with analogous facilities which support other knowledge structures? We elaborate on these questions in the following sections.

2.2 Issues in explaining choices

What makes for a 'convincing' justification of a choice? Clearly, we need to do better than simply display the value function and its arguments. Referring again to JESQ's domain, imagine an explanation of the form:

Copying the dataset to tape is your best option because your value function is $v(x_1, x_2, x_3, x_4) = .2v_1(x_1) + .4v_2(x_2) + .1v_3(x_3) + .3v_4(x_4)$, where x_1 is additional operator time, x_2 is additional turnaround time, x_3 is additional user time, and x_4 is additional material cost, and for this option $(x_1, x_2, x_3, x_4) = (10,10,0,5)$, which maximizes v over all available alternatives.

We can identify several problems with this justification. First, no effort is made to appeal to intuition; the explanation does not associate components of the value function with concepts such as 'desirability' and 'importance'.

Second, the explanation refers only to the most detailed attributes upon which the decision is based. For example, the concepts of 'turnaround time' and 'user work' are both associated with the higher-level concept 'user satisfaction', but this sort of information is omitted from the explanation. It might even have been possible to talk solely in terms of 'user satisfaction', omitting any reference to its more detailed supporting measures.

Third, the explanation takes the 'brute force' approach of elucidating *all* the attributes that underlie the decision, while it is most likely that only one or perhaps a few truly served to distinguish the chosen alternative from its closest contenders. For example, deleting a dataset, like copying one to tape, involves little additional material cost. Thus, the attribute 'additional material cost' plays a relatively minor role in distinguishing these two alternatives in terms of their overall relative desirability.

A related objection is that the order in which attributes are mentioned is essentially arbitrary from the user's viewpoint; he cares little about the order of the arguments in the value function's parameter list. What the user wants to know is which factors in the decision most strongly recommend the chosen alternative.

Fifth, the explanation makes no reference to the presumed conditions under which the value function is applicable. Is turnaround time *always* the most important attribute? Or are there circumstances under which it would be weighted differently? It is critical to elucidate such conditions if

explanations are to be convincing and if users are to be able to gain enough insight into the model's operation to correct it.

We might also enhance the explanation in other ways, such as reassuring the user that value functions built by several operations managers reflect the same preferences, or by providing a higher level description of the basic goals of the installation, or by substituting qualitative descriptions for quantitative values (e.g., 'lots of time' rather than '10 minutes').

The essential point is that justifications for choices should be more than displays of the models and parameters that determine them. How can we provide these capabilities? What knowledge stores beyond the value function are required? How should this additional knowledge be represented? A goal of the proposed thesis is to provide a computational framework for generating justifications which are not subject to the above-mentioned criticisms. Some preliminary ideas are presented in chapter V.

2.3 Issues in refining choice knowledge

Suppose the user is unconvinced by the system's justification for a choice. Should he be forced to rebuild the underlying model from scratch? Should he call upon a programmer to change the model? Existing work on automating initial acquisition fails, by itself, to provide the solution. These facilities are intended to guide users through the systematic capture of utility functions, not through their incremental modification.

In contrast, the purpose of a refinement facility is to make it convenient for users to repair only those portions of the model which fail to reflect reality. Toward this end, a refinement facility should:

- provide immediate feedback regarding the effects of proposed changes;
- help users to distinguish probable from improbable causes for erroneous choices;
- promote the reliability of newly-captured information;
- isolate subproblems for correction;
- infer new model parameters based on logical relationships between existing parameters and prompt the user for missing details; and
- help the user to identify modifications which will achieve the behavior he desires.

This list is surely not exhaustive, but communicates the flavor of the capabilities we should expect from the refinement facility. Each of the above capabilities is addressed in more detail in chapter V.

In short, the goal of refinement is to maximize the reliability of model repairs while minimizing user effort. But how should this be accomplished? What addition knowledge is required? How should it be represented? The second central goal of the thesis is to answer these questions.

2.4 Issues of integration with other decision-making paradigms

Our discussion of research issues in explanation and refinement of choice knowledge has more or less focussed on the AMVM as a 'top-level' structure, existing in isolation for the sole purpose of helping users to make choices. But there is the important issue of how facilities for explaining and refining choice knowledge should be integrated with analogous facilities that support other knowledge structures with which the AMVM may coexist. If we use an AMVM as a conflict resolution algorithm in a production system, for example, how might the facilities which support the explanation and acquisition of rules be integrated with the facilities for explaining and refining choices among them? What are the interfaces between such facilities? It is important to address such

questions if we are to take advantage of decision-theoretic models in traditional AI systems and other intelligent system architectures.

Chapter IV: Background and Relationship to Previous Work

The proposed work is interdisciplinary, addressing goals which are relevant to the artificial intelligence and decision analysis communities. In this section we describe relevant research organized in terms of the following categories:

- decision analysis
- integration of decision-theoretic models and architectures commonly associated with AI
- models of choice commonly used in AI systems
- automated explanation
- automated knowledge acquisition
- integrating explanation and refinement
- user modelling

Of interest are foundational works related to the proposed thesis, research efforts from which we may borrow ideas, and efforts which contrast with the goals of our work.

1. Decision analysis

The framework of decision analysis is concerned with choosing among alternatives in an uncertain environment. The framework provides for the systematic treatment of the *utilities* of outcomes resulting from alternative courses of action and their associated *probabilities*. Keeney (1982) describes decision analysis as 'a formalization of common sense for decision problems which are too complex for informal use of common sense', and more technically as 'a philosophy, articulated by a set of logical axioms, and a methodology and collection of systematic procedures, based upon those axioms, for responsibly analyzing the complexities inherent in decision problems'. In this section we cite the major references concerning the theoretical foundations of decision analysis and its methodology, and better-known examples of its application. For a concise and informative overview of the field, see (Keeney 1982).

Foundational works on utility theory include (von Neumann & Morgenstern 1947), (Savage 1954), (Luce & Raiffa 1957), and (Pratt et al. 1964, 1965). A foundational reference on *multiattribute* utility theory is (Keeney & Raiffa 1976). In particular, models of *value functions* addressing multiple objectives may be found in (Debreu 1960), (Luce and Tukey 1964), (Krantz 1964), (Krantz et al. 1971), (Dyer & Sarin 1979), (Kirkwood & Sarin 1980), and (Keelin 1981). A more concise review of multiattribute decision making may be found in (Spronk & Zionts 1984).

Works which address methodological considerations in applying decision theory to practical problems include (Raiffa 1968), (Schlaifer 1969), (Tribus 1969), (Winkler 1972), (Brown et al. 1974), (Keeney & Raiffa 1976), (Moore & Thomas 1976), (Kaufman & Thomas 1977), (LaValle 1978), and (Holloway 1979). In particular, the systematic assessment of utility functions is addressed in (Fishburn 1967), (Huber 1974), (Keeney & Raiffa 1976), and several other works. For an informative review, see (Farquhar 1984).

Reported applications of decision analysis span a diverse set of domains and decisions, including the examination of corporate policy (Keeney 1975), evaluation of capital investment options (Magee 1964), budget allocation (Keefer & Kirkwood 1978), credit application evaluation (Stillwell et al. 1980), medical decision making (Krischer 1980), hurricane seeding (Howard et al. 1972), metropolitan airport development (de Neufville & Keeney 1972), fire protection (North et al. 1975), school busing (Edwards 1980), oil tanker standards (von Winterfeldt 1982, Ulvila & Snider 1980), nuclear waste management (Lathrop & Watson 1982), commercialization of solar photvoltaic systems (Boyd et al. 1982), siting of energy facilities (Keeney 1980, Sarin 1980), and many others.

The relationship of this work to ours should by now be clear; we are interested in using decision-analytic models and techniques in intelligent systems and in providing the necessary supporting facilities. References to specific results of relevance are cited in context throughout the proposal.

2. Integrating decision theory and AI techniques

We are hardly the first to advocate the use of decision theory in the context of architectures commonly associated with AI; in part, the proposed thesis is motivated by previous efforts which combine the two historically distinct paradigms to achieve effective performance, but lack the supporting facilities that we will attempt to provide.

For example, Coles et al. (1973) used utility theory to evaluate robot plans as a means for coping with uncertainty. Based on the accumulated expected costs of executing the steps of various hypothetical plans, the robot Jason can evaluate the relative merits of direct action using potentially unreliable sensors. Jacobs et al. (1973) performed experiments based on similar ideas. Feldman & Sproull (1975) used decision theory to direct the application of planning operators in an implementation of the monkey and bananas problem. White & Sykes (1986) used a generalization of multiattribute utility theory as the basis for conflict resolution in a rule-based system. Langlotz et al. (1986) explored the use of decision theory to justify heuristics in the context of MYCIN (Shortliffe 1976). Using plots and calculations generated by an automated decision-making tool, decision-theoretic insights of practical use to the knowledge engineer were obtained. Langlotz et al. (1985) also describe a cancer therapy planning system which generates a small set of plausible plans, simulates them to predict their possible consequences, and uses decision theory to rank them. Slagle & Hamburger (1985) describe an interactive planning system that uses decision-theoretic models to rank competing plans for allocating military resources. O'Leary (1986) discusses the use of multiattribute decision theory in expert systems for financial accounting decisions.

This list is no doubt incomplete, but it should be clear that researchers have recognized the potential for using decision theory as a model of choice in intelligent systems. For further discussion of the use of decision theory in expert systems, see (Keeney 1986). Farquhar (1986) reviews some additional applications of utility theory in AI contexts.

3. Models of choice in AI systems

While the previous section indicates that decision theory has received some attention in AI contexts, the employment of formal models of choice in intelligent system architectures represents a departure from more common approaches to choices in AI systems, including hand-crafted explicit selection schemes, hand-crafted implicit selection schemes, and languages for encoding knowledge about choices.

3.1 Hand-crafted explicit selection schemes

A popular approach to modelling choice in intelligent systems is to construct *ad hoc* models which are tailored to particular domains. This is the approach of, for example, some expert systems in medical management (e.g., Kastner 1983, Clancey 1984) and spectral analysis (e.g., Ferrante 1985). The approach essentially involves adopting a 'mindset' for choosing among alternatives in the domain at hand and reflecting this mindset in a computational model that serves as a basis for computing and explaining choices. The factors underlying choices are explicitly represented to facilitate acquisition and explanation.

Rennels et al.(1987) showed that some seemingly *ad hoc* models could actually be viewed as restricted decision-theoretic models. The authors point out that these models are capable of

producing more focussed explanations because the strategies they implement already entail strong assumptions about their respective domains. Generally speaking, however, the hand-crafted approach encompasses some important limitations.

First, domain-specific models often lack justification with respect to a well-formed underlying theory. As we discussed in previous sections, this means that we lack a basis for understanding their (mis)behavior, and so we are provided with little direction in building and maintaining them.

Second, the implicit operational assumption is made that the user and the intelligent system share a common view of decision-making. As this may not be the case for all users, explanations which rely on this commonality may be opaque, or worse still, misleading.

Third, the general approach of hand-crafting explanation and acquisition facilities for particular domain choices is a labor intensive one. In building a new expert system that employs a model of choice, we are essentially required to either cast a domain into one of the existing restricted models in order to use an existing framework for acquisition and explanation, or to develop another set of facilities that better suits the domain. Since we can't necessarily identify the assumptions that underlie a particular hand-crafted model it might be difficult to select an existing framework to suit a particular domain.

Our work is distinguished from the hand-crafted approach to modelling choice in that we advocate a general, well-formed theory of choice which is applicable across a variety of domains.

3.2 Hand-crafted implicit selection schemes

At the other end of the spectrum are *implicit, ad hoc* models of choice. Heuristic evaluation functions in AI game-playing programs (Nilsson 1980) provide a good example. Since these models do not explicitly represent the factors that underlie choices, they are effectively 'black boxes', with no basis for automating knowledge acquisition or explanation (beyond justifications of the form, 'this is the best move'). Most production system conflict resolution algorithms (e.g., production order, special case, distinctiveness, and recency rules (McDermott & Forgy 1978)) exemplify this type of choice model, selecting instantiations using hard-coded algorithms.

Our work is distinguished from these efforts in that we are dealing with explicit models of choice, with transparency and ease of evolution being among our principal concerns.

3.3 Languages for encoding choice knowledge

Another approach to modelling choices in AI systems involves providing *computational formalisms* for encoding knowledge about selecting among knowledge-level objects. These are essentially languages for expressing knowledge about choices which say nothing about the nature or elements of choice knowledge. The formalisms may be categorized in two groups: procedural and declarative.

FCL (Friedman 1985) provides an example of the procedural variety where the objects of choice are productions in a production system. The language provides constructs such as functions, function calls, and sequencing for controlling invocation. A similar example, Georgeff (1982) proposed a general production system architecture that allows procedural control knowledge to be directly represented and used. While such approaches provide more flexibility in rule-writing, they suffer from some of the same problems as the above-described hard-coded algorithms. First, the languages do not implement any sort of theory of choice; the approach is one of 'hacking' to achieve the desired results. Second, maintenance is still primarily a programming task rather than a process of modifying a well structured repository of knowledge. Third, since procedural control leaves implicit the knowledge that underlies the sequencing of productions, this knowledge cannot be elucidated for the user in an explanation. Our work is distinguished from this approach in that

we advocate the implementation of a formal theory of choice which takes as input a set of parameters rather than arbitrary code. In part, this is to make the knowledge underlying choices accessible for explanation and modification. Clearly, this knowledge is inaccessible under the procedural language-oriented approach.

The declarative variety of language-oriented control is exemplified by meta-rules (Davis 1976, Davis 1980). Treating conflict resolution as a problem-solving task itself, meta-rules direct the invocation of object-level rules which encode domain knowledge. Under this approach, the knowledge underlying selection of object-level knowledge is explicitly represented and may be domain-dependent. This has considerable advantages with regard to maintainability and explainability; since choice knowledge is explicitly represented, both of these tasks are facilitated. But meta-rules still represent a computational formalism for encoding choice knowledge, not a theory of choice that assigns precise semantics to particular models and their parameters. In addition, we can identify some problems with the meta-rule approach. First, meta-rules are economical in the sense that they make use of the same machinery that supports object-level problem solving, but there is no reason to assume that this machinery is ideal or even appropriate for making choices. Second, the meta-rule approach leaves open the question of what the 'top-level' conflict resolution algorithm should be; That is, it is unclear as to how a system would select among the highest level of meta-rules encoded. Third, some control knowledge still resides in the inference engine under the meta-rule approach, and it seems somewhat misguided to arbitrarily distribute control knowledge in two places, one of which (the inference engine) is opaque. Also, it seems misguided to house control knowledge with object-level knowledge in the first place.

In summary, our approach deviates from that of computational formalisms for encoding choice knowledge in that we advocate some theory of choice wherever explicit models of choice are needed. In the case of procedural languages for making choices, we also diverge in our effort to support transparency and knowledge acquisition.

4. Automated explanation

It is by now generally agreed that explanation is a fundamentally important supporting capability for intelligent systems. A well-known study by Teach & Shortliffe (1981), for example, revealed that high quality explanation capabilities were the most important requirement for an acceptable clinical consultation system, concluding that a 'system should be able to justify its advice in terms that are understandable and persuasive ... A system that gives dogmatic advice is likely to be rejected'. Explanation has become a central topic of research, with experiments in a set of domains as diverse as the blocks world (Winograd 1972), medicine (Davis 1976, Aikens 1980, Clancey 1981, Swartout 1983), complex physical machinery (Forbus & Stevens 1981, Stevens 1981, de Kleer & Brown 1984, Weld 1984), game playing (Berliner & Ackley 1982), and financial planning (Kosey & Wise 1984), just to name a few. We will not give an exhaustive overview of explanation research here; rather, we focus on those efforts which contribute to or contrast with the explanation component of the proposed thesis.

4.1 Justifying choices

Of primary relevance to the explanation component of this work are research efforts involving the justification of complex choices between competing alternatives. Perhaps most relevant is the work of Langlotz et al. (1986) on generating explanations from single-attribute utility functions. Using decision trees, information attached to nodes in the form of frames, and qualitative mathematical reasoning, the authors generate intuitively appealing justifications for choosing particular treatments in the medical domain. The explanation portion of our work will be similar to theirs in its use of traditional decision-theoretic structures coupled with supplementary knowledge stores to generate explanations. Our work will differ greatly in our focus on multiattribute value functions and the

associated use of objective hierarchies, and primarily in the interactive, integrated nature of explanation and refinement. Langlotz et al. did not address refinement in their work. As a probable result, their explanations avoid presentation of any quantitative values or underlying model structure. This also represents a diversion for us, in that we must provide a clearer window into the model itself in order for users to correct it.

In contrast, BLAH (Weiner 1980) generates explanations for choices based on a very limited, *ad hoc* model which is restricted to reasoning between two alternatives. While we will borrow functional aspects of explanation from BLAH, our work will necessary be different because we are justifying choices which are based on a much more powerful underlying model. In addition, BLAH employs certain strategies with which we patently disagree on the basis of the data we have collected from human beings.[25] For example, Weiner asserts that 'All reasons for a decision are, of course, part of the explanation of it'. Our data clearly refutes this.

Also relevant are the hand-crafted models of choice which Rennels et al. (1987) showed to be restricted forms of decision-theoretic models (Kastner 1983, Clancey 1984). These models were developed with explanation as a primary design goal, and might provide some useful lessons regarding the provision of justification for choices. Again, our work differs from these efforts in that we address more general, formal models, without making restrictive assumptions that limit the use of our facilities to one or to a few domains.

4.2 Explaining the results of other quantitative models

Also relevant is work on explaining quantitative models outside the realm of value-based choices. For example, there exist systems which explain diagnostic conclusions based on probabilistic models (e.g., (Ben-Bassat et al. 1980)). While these systems produce very quantitative explanations, and focus on diagnostic conclusions rather than on the choice of problem-resolving actions, we can make use of some of their abstract presentation strategies such as ordering evidence by its importance to decisions (Reggia & Perricone 1985) and separating evidence which supports conclusions from evidence which denies them (Speigelhalter & Knill-Jones 1984). Our approach to explanation, which attempts to incorporate the elements of human communication, greatly differs from these efforts which provide tabular presentations of quantitative data.

The ROME system (Kosey & Wise 1984), which answers queries about financial spreadsheets, also provides some insights which are useful to us in the proposed work. In formulating explanations, ROME employs strategies such as distinguishing relevant parts of the underlying model from irrelevant parts, identifying significant variables in particular situations, and translating quantitative values into qualitative ones for presentation. Our analysis of the justifications for choices offered by human beings[25] motivates similar strategies in our work. Our work differs not only in the underlying model being elucidated, but in our integration of explanation with refinement, our focus on explaining the underlying model to the user as well as specific results, our use of multiple models organized by user, and other characteristics.

4.3 Summary

The explanation component of UTIL will draw upon previous results in explanation contributed by a variety of authors. But UTIL will be unique in several respects, representing the first integrated system for explaining and refining decision-theoretic models, and the first domain-independent effort to explain the very general and commonly employed AMVM.

[25] A detailed discussion appears in chapter V.

5. Automated knowledge acquisition

Automated knowledge acquisition is another fundamental supporting capability for intelligent systems. As the acquisition of knowledge is often described as the primary bottleneck in intelligent system development (Waterman 1986), work on its automation has become a popular research area. Many different approaches to automating the construction and improvement of intelligent systems have been proposed over the past several years, ranging from the interactive transfer of expertise (Davis 1976) to machine learning (Michalski et al. 1983, 1986). Our approach to the interactive refinement of value functions is clearly an instance of the former. As in the case of explanation, it is not our purpose to review the field of knowledge acquisition here; rather, we cite approaches which are specifically related to or counter to the proposed work.

5.1 Initial acquisition of decision-theoretic models

In chapter III we distinguished two distinct phases of development for decision-theoretic models: *initial acquisition* (building the model from scratch, including problem structuring, model selection, and parameter assessment) and iterative *refinement* (incremental problem restructuring and parameter tuning). As previously mentioned, the systematic assessment of utility functions is a relatively mature topic of research (e.g., see (Farquhar 1984) for a review). Many methods have been implemented in interactive computer programs, for example, (Keeney & Sicherman 1976), (Nair & Sicherman 1979), (Novick et al. 1980), (Schlaifer 1971), (Seo et al. 1978), (Klein et al. 1982), (Weber 1985), and (von Nitzsch & Weber 1986). The existence of such programs renders it reasonable for us to focus our efforts on refinement, without worrying about initial acquisition as well. However, existing methods for initial acquisition should provide a number of ideas which we can adapt for refinement.

5.2 Refinement of decision-theoretic models

Refinement involves the incremental modification of the utility function over time to correct uncovered errors and to reflect new preferences as the system evolves. One approach, analogous to the role of machine learning in expert systems, involves learning the parameters of a utility function. For example, Madni et al. (1985) describe a system for learning the weights in an additive utility function. A related approach, adaptive utility theory (Cyert 1975, Cohen 1984), involves methods for converging on a precise utility function by updating parameters based on experience. Our work is essentially unrelated to these efforts.

We are more interested in approaches involving the *interactive* refinement of preferences, and there has been relatively little work in this area. Lehner (1985) developed a rule-based system for assessing utility function parameters which supports a limited version of refinement in that the user may choose to reassess only certain parameters over time, but it does not (as far as we can tell) address the issues of chapter III. Wellman (1984) describes a very interesting approach to initial acquisition that might be extended for refinement. The system selects a form for the utility function by proving the theorems which justify using such forms. But this system too does not aid the user in refinement *per se*, and assumes that the user is knowledgeable about utility theory. Our work will build upon these efforts by providing a framework for modifying (restricted) utility functions which avoids resorting to initial acquisition methods (i.e., rebuilding models from scratch).

5.3 Summary

There exist several automated facilities which support the initial acquisition of decision-theoretic models, and we shall assume that these may be employed to develop the value functions which UTIL explains and helps to refine. We will also draw upon the applicable elements of these methods in developing UTIL's refinement facilities. Our inventory of systems for interactively refining

decision-theoretic models reveals that much work remains to be done, and UTIL will help to fill this gap.

6. Integrating explanation and refinement

This work is greatly influenced by Teiresias (Davis 1976), a vehicle system constructed to explore applications of meta-level knowledge (Davis & Buchanan 1977). Our basic approach to designing UTIL's explanation and refinement facilities, particularly their interrelationship, is inspired by Teiresias' analogous facilities for rule-based systems. Teiresias encompasses a view of knowledge acquisition as the interactive transfer of expertise (involving a human expert and a program). Loosely speaking, UTIL may be viewed as a 'Teiresias for decision-analytic models'.

UTIL will share several themes with Davis's work. In particular, we will build upon Davis's notion of capturing and verifying new information in the context of specific situations and upon the use of explanation facilities as a window into system behavior which facilitates modification.

7. User modelling

The proposed work is related to the field of *user modelling* (e.g., see (Kass & Finin 1987) for a review), in two ways. The first, briefly discussed by Langlotz et al. (1986), involves the employment of an explicit model of the user which is used to tailor justifications for choices to his tastes. We do not plan to do any work in this area.

The second link to user modelling concerns the use of utility functions *as* user models. Since a utility function is a representation of a particular user's preferences in a specific decision-making situation, we may regard a collection of such functions as a model of the expert. Chapter V describes such a structure (called a *user profile*) which is used by UTIL to make analogies with other decision problems and to reference the opinions of other decision makers in an explanation. Some work in user modelling encompasses the encoding of user preferences (e.g., Rich 1979, 1983), but the approach is one of maintaining sets of attributes which describe the user in general terms, rather than sets of objectives (and their interrelationships) which pertain to particular decisions. We will continue to examine the user modelling literature to identify other ideas and themes which may be potentially useful in UTIL.

Chapter V: Preliminary Work

We have done preliminary work toward UTIL's development in essentially four related areas, using JESQ's domain (chapter II) as a vehicle throughout:

1. We have formulated a simplified but realistic AMVM for choosing among competing space-freeing actions;
2. We have sketched some alternative architectures for reimplementing JESQ, each of which includes this AMVM, in order to illustrate its potential use in various architectural contexts;
3. Toward the design of the explanation component, we have collected justification-oriented dialogs from computer operators and used these as the basis for the preliminary design of explanation commands and responses. We have also done some preliminary work on the design of mechanisms to support these responses.
4. Toward the design of the refinement component, we have outlined some capabilities and mechanisms to support them.

The chapter is organized as follows. Section 1 describes an AMVM and supporting analysis for the queue space management problem. Three distinct architectures which might employ this model are described in section 2, and their implications for the required level of modularity in UTIL are discussed. Section 3 demonstrates the usual ease with which new alternatives in JESQ's domain might be introduced. UTIL's proposed explanation facilities and the rationale underlying their design are described in section 4. Section 5 contains an analogous presentation of UTIL's refinement facilities.

1. A sample AMVM: JESQ revisited

In this section we formulate a simplified version of JESQ's choice problem in terms of the AMVM. This serves three purposes: (i) It provides an example with which to illustrate the issues and preliminary techniques introduced in this chapter; (ii) It elucidates some of the analysis involved in casting decisions in terms of the AMVM;[26] and (iii) It allows us to become more specific about the sort of model of choice that we claimed would have enhanced JESQ.

We will not recast the complete domain here, for that will make hand-simulated examples in later sections difficult to understand. Rather, we'll address only the following decision-making situation: Suppose that a user generates a large dataset that is to be printed on a printer which is not currently working, and that a choice between the following alternative plans for manipulating that dataset must be made in order to free some space on the operating system queue:

1. **copy**: copy the dataset to tape and print it later when the requested printer has been repaired.
2. **expensive-printing**: print the dataset on a faster printer which uses expensive forms.
3. **cheap-printing**: print the dataset on a slower printer which uses lower quality forms than those requested by the user.
4. **delete**: delete the dataset from the queue.
5. **install**: temporarily install a duplicate of the requested printer for use until the existing printer has been repaired.
6. **fiche**: deliver the dataset to the user on microfiche.

[26] Note, however, that this formulation lacks the rigor of an 'industrial strength' analysis. For example, we developed the component value functions from intuition rather than using standard assessment techniques. See Keeney (1982) for an informative overview of the methodology of decision analysis. Keeney and Raiffa (1976) provide more detailed illustrations.

7. **cards**: deliver the dataset to the user on punched cards.
8. **dasd**: transfer the dataset to the user's private disk storage so that he can later transfer it back to the queue for printing after the printer has been repaired.

Note that some of these may not be feasible in certain implementations of JES/MVS, some of them are configuration dependent, and some (e.g., install) are just plain silly.[27]

1.1 Structuring objectives

First, we structure the objectives of installation management which underlie choices between operational heuristics, using an *objectives hierarchy* (Keeney & Raiffa 1976). The idea is essentially to capture the hierarchical nature of objectives in a corresponding hierarchical structure where the satisfaction of a given objective is measured in terms of the satisfaction of its component objectives (i.e., children in the hierarchy). Primitive objectives (at the bottom of the hierarchy) are measured by their associated *attributes*. An attribute should be both comprehensive (i.e., indicative of the level to which the associated objective is achieved) and measurable (i.e., the decision maker can specify preferences for different possible levels of the attribute). Of course, this initial problem structuring would be accomplished by decision analysts and knowledge engineers in the initial stages of development of an intelligent system, with no help from UTIL. Initial parameter assessment might also be performed with the help of humans, or by an automated facility outside UTIL.[28]

Recall that the perceived effectiveness of space management actions is judged in terms of several objectives (see chapter II), including maximizing the impact on the JES queue space, maximizing the convenience of the operator, maximizing the satisfaction of the user community, minimizing wasted material costs, and maximizing the speed of actions. These, in turn, were defined in terms of more detailed objectives. We want to slightly simplify this set of objectives here for ease of presentation.

Let us assume that a single dataset is to be manipulated, so that the objective 'maximize impact' becomes irrelevant. This will limit the number of alternative actions to the mentioned eight. We will thus choose among competing space-freeing alternatives based on four objectives: 'maximize operator convenience', 'maximize user satisfaction', 'minimize material costs', and 'minimize space clearing time'. In our simplified formulation, 'maximize operator convenience' may be considered to directly correspond to 'minimize the amount of additional time the operator spends performing an action beyond that originally required' (i.e., before the printer broke down and created the problem), measured in terms of the attribute *minutes*. 'Maximize user satisfaction', on the other hand, might be decomposed into more detailed, lower-level objectives such as 'minimize turnaround time in excess of that originally required' (measured in terms of the attribute *minutes*) and 'minimize difference between the output medium originally requested and the output medium associated with the chosen action', measured in terms of a subjective index which assigns 0 to forms most different from those requested and 1 to precisely those requested and forms which are better than those requested. Note that we could have instead measured 'user satisfaction' directly in terms of some subjective index (e.g., very-, marginally-, dis-satisfying), but in this case more detailed (less opaque) objectives are indeed available. 'Minimize material costs' (in excess of those initially requested) can be directly measured in terms of the attribute *dollars*. 'Minimize queue space clearing time', the time that it takes for the dataset to actually exit the queue (excluding other processing) can be directly measured in *minutes*. Pictorially, we have the objectives hierarchy of Figure 1 for our sample problem.

---

[27] These have been included for purposes of exposition. In an actual decision analysis, such alternatives would be omitted from the start.

[28] See chapter IV for references.

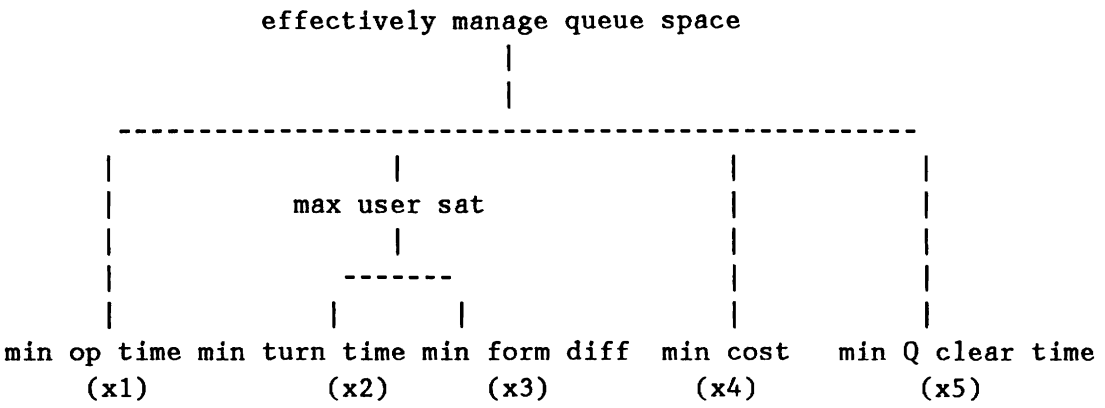

Figure 1: Objectives hierarchy for simplified queue space problem

Our attributes are x_1 = additional operator time in minutes, x_2 = additional turnaround time in minutes, x_3 = difference in form (subjective index), x_4 = additional cost in dollars, and x_5 = (absolute) queue space clearing time in minutes.

1.2 Assessing the impact of alternative plans

Next, we assess the impact of each alternative and represent it in a vector of values for our attributes, forming a sort of 'profile' for the alternative. Copy, for example, is represented (10,34.2,1,1,15.1) according to the following analysis. It takes the operator approximately 10 minutes to make sure a tape is mounted, start the copy process, walk the tape to/from the tape library, etc., so x1 = 10.[29] Fixing the printer, and copying and restoring the users job will take around say, 34.2 minutes on average, thereby increasing his turnaround time by that amount (so x2 = 34.2).[30] Since the copy plan encompasses bringing the dataset back onto the queue and printing it (after the printer has been repaired), the user receives his data on the output medium requested (thus x3 = 1). While tapes and tape drives are reusable, we amortize the cost of copying a job at about $1 per job, so x4 = 1. We assign x5 = 15.1 because the dataset stays on the queue until the copy operation has been completed (approximately 15.1 minutes). By similar analyses, we obtain the vectors shown in Table 1.

Alternative (Plan)	x1	x2	x3	x4	x5
copy	10.0	34.2	1.0	1.00	15.1
expensive-printing	0.1	0.0	1.0	100.00	25.0
cheap-printing	0.1	10.0	0.8	0.00	40.0
delete	0.0	infinity	0.0	0.00	0.1
install	180.0	180.0	1.0	5000.00	210.0
fiche	0.1	20.0	0.2	70.00	50.0
cards	0.1	15.0	0.1	20.00	45.0
dasd	0.1	32.1	1.0	0.50	1.0

Table 1: Attribute vectors for alternatives

[29] In an actual analysis we might collect historical data to compute these averages.

[30] Lest there be confusion because x2 > x1, the operator does not have to stand at the tape drive while file transfer proceeds. Thus, it is reasonable that only 10 minutes of the operator's time is spent in the execution of this alternative even though the copy process takes longer. The same explanation describes why it is reasonable that x5 > x1.

Several assumptions were made in formulating Table 1, including:

1. Our constant dataset size is 1 million lines. In an actual implementation, various attribute values (e.g., the cost of expensive paper) would be computed as a function of the number of lines in the dataset. Here, they are constants.
2. Attribute values which might be recorded as negative (i.e., in cases where the alternative actions are *better* with respect to the associated objectives) are recorded as zero to reflect the wording 'in excess of' in our formulation of objectives. Equivalently, we might have recorded them as negative and treated them as zero in the component value functions described later. This is done because the operators would have precisely honored the user's request for resources had the printer not been down, rather than attempted to *optimize* output processing. They (and hence we) assume that the user will not be made any happier, for example, by expensive paper if that paper was not requested. Thus, setting these attribute values to zero negates the potential positive utility derived from 'negative excesses'. We would not want these negative excesses to offset true (positive) excesses in choosing the best alternative.
3. Alternatives which involve reestablishment of the dataset on the queue for requested printing (e.g., dasd, copy) assume approximately 30 minutes waiting time for printer repairs. In an actual implementation, this value would be input on a situation by situation basis and the affected attribute values computed accordingly.

1.3 Encoding preferences

Next, we determine the operational policy (preferences) of installation management regarding selection among alternatives and encode this policy in the value function. Conversations with operators suggest that the attributes satisfy the independence assumptions which justify employing the additive model. But what is the relative importance of these attributes with respect to choices between competing alternatives?

To some extent, the relative importance of each attribute seems to depend on factors outside the model, primarily the severity of the current queue space situation. If, say, 25% of queue space remains free, then the installation is willing to sacrifice some more of an operator's time in order to provide better service (i.e., more user satisfaction). On the other hand, if only 5% of queue space remains free, there is significant danger that the target system may 'crash', and so, an individual's satisfaction is traded off for more judicious use of the operator's time. Does this imply that 'amount of queue space left' should itself be represented as an attribute, with all others conditionally dependent on it? We think not; in fact this would be very unnatural, for the amount of space left at the time the decision is made fails, in itself, to reflect any coherent objective. Rather, the amount of space left suggests the tradeoffs between the identified attributes, and thus serves as an index into a particular value function; different value functions may be appropriate under different levels of space left. For the time being, let us assume that the situation is still under control (i.e., there is sufficient space left to avoid panic) ; we will develop a value function which reflects this assumption.

Using standard acquisition methods, we might determine that the weights for attributes are distributed as in Figure 2. This distribution is consistent with our assumptions about tradeoffs when the situation is not critical.

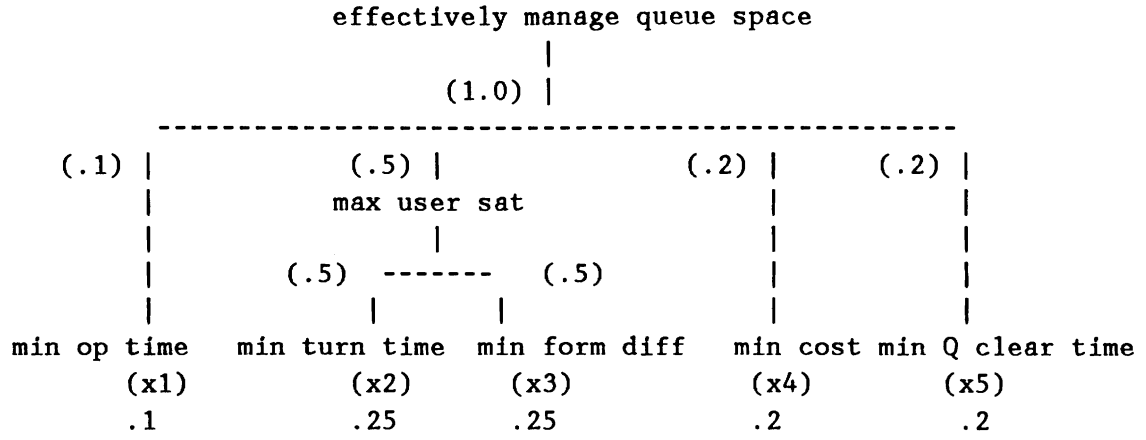

Figure 2: Initial distribution of weights

We thus have the following value function for assessing queue space management actions:

```
v(x1,x2,x3,x4,x5)
= .1*v1(x1) + .5(.5*v2(x2) + .5*v3(x3)) + .2*v4(x4) + .2*v5(x5)
= .1*v1(x1) + .25*v2(x2)   + .25*v3(x3) + .2*v4(x4) + .2*v5(x5)
```

where v_i is the component value function for the ith attribute. Note that the hierarchy allows for the isolated assessment of decision problems at different levels of abstraction, with weights summing to 1 at any given level.[31] At the highest level, user satisfaction accounts for 50% of the decision, with the remaining objectives accounting for the remaining 50%. As for user satisfaction itself, the only decomposed objective, turnaround time accounts for 50% of the user satisfaction assessment, with difference in form accounting for the remaining 50%. Multiplying the weights as implied by the hierarchy, we arrive at values of 25% and 25% for these detailed attributes of user satisfaction, without requiring the user to directly specify these.

Again using existing automated facilities, we might capture the component value functions listed in Table 2 for our attributes. Note that for objectives which are to be maximized (minimized), the value functions assign 0 to the lowest (highest) attribute values and 1 to the highest (lowest) values, with all others lying in between.

[31] As will be shown, this property is exploited in both explanation and refinement.

v1: excess operator time (minutes)

x1	v1(x1)
x <= 5	1.0
5 < x <= 20	0.5
20 < x <= 60	0.3
x > 60	0.0

v2: excess turnaround time (minutes)

x2	v2(x2)
x <= 5	1.0
5 < x <= 10	0.8
10 < x <= 20	0.6
20 < x <= 40	0.4
40 < x <= 60	0.2
x > 60	0.0

v3: difference in form (subjective index: 0 -> 1)

v3(x3) = x3 for all x3

v4: excess material cost ($)

x4	v4(x4)
x <= 10	1.0
10 < x <= 40	0.8
40 < x <= 80	0.6
80 < x <= 100	0.4
100< x <= 150	0.2
x > 150	0.0

v5: space clearing time (minutes)

x5	v5(x5)
x <= 10	1.0
10 < x <= 20	0.8
20 < x <= 30	0.6
30 < x <= 40	0.4
40 < x <= 60	0.2
x > 60	0.0

Table 2: Component value functions

1.4 Evaluating alternatives

Finally, we evaluate and compare our alternatives as depicted in Table 3. The value function reflects the information in Figure 2. The consequences of alternatives reflect the contents of Table 1. Component value function evaluation reflects Table 2.

```
v(dasd) = v(.1,32.1,1,.5,1)
= .1*v1(.1) + .5(.5*v2(32.1) + .5*v3(1)) + .2*v4(.5) + .2*v5(1)
= .85

v(expensive-printing) = v(.1,0,1,100,25)
= .1*v1(.1) + .5(.5*v2(0) + .5*v3(1)) + .2*v4(100) + .2*v5(25)
= .8

v(cheap-printing) = v(.1,10,.8,0,40)
= .1*v1(.1) + .5(.5*v2(10) + .5*v3(.8)) + .2*v4(0) + .2*v5(40)
= .78

v(copy) = v(10.0,34.2,1.0,1.00,15.1)
= .1*v1(10) + .5(.5*v2(34.2) + .5*v3(1)) + .2*v4(1) + .2*v5(15.1)
= .76

v(delete) = v(.1,infinity,0,0,.1)
= .1*v1(.1) + .5(.5*v2(infinity) + .5*v3(0)) + .2*v4(0) + .2*v5(.1)
= .5

v(cards) = v(.1,15,.1,20,45)
= .1*v1(.1) + .5(.5*v2(15) + .5*v3(.1)) + .2*v4(20) + .2*v5(45)
= .475

v(fiche) = v(.1,20,.2,70,50)
= .1*v1(.1) + .5(.5*v2(20) + .5*v3(.2)) + .2*v4(70) + .2*v5(50)
= .46

v(install) = v(180,180,1,5000,210)
= .1*v1(180) + .5(.5*v2(180) + .5*v3(1)) + .2*v4(5000) + .2*v5(210)
= .25
```

Table 3: Evaluation of alternatives for the sample problem

This formulation of the problem imposes the following ordering on alternatives: dasd > expensive printing > cheap printing > copy > delete > cards > fiche > install.

2. Some AMVM-based architectures for the sample problem

In this section we discuss some of the potential architectural contexts in which the sample decision model of the previous section might be usefully implemented. Our purpose here is (i) to make more explicit our somewhat vague references to the use of decision-theoretic models as 'top-level' and 'component machinery' in intelligent systems in prior chapters, (ii) to show how JESQ might be enhanced by incorporating the AMVM in a similar architecture, and (iii) to provide a basis for

defining a common interface between some popular intelligent system architectures and decision-theoretic models, so that we may refer to these interfaces in our discussion of explanation and refinement.

We proceed as follows. First, we discuss the component abstract tasks of intelligent process control. These may be performed by the operator, the intelligent system, or by some cooperative arrangement involving both.[32] The next few sections sketch various ways in which these tasks might be implemented by integrating the AMVM with some standard architectures, and examine the implications for constructing transportable explanation and refinement facilities for the AMVM.

2.1 Component tasks in intelligent process control

Intelligent process control may be viewed as encompassing the following subtasks:

- *monitoring*: An intelligent agent (operator or system) must obtain the values of target system state variables in order to (i) detect problem conditions and to (ii) ascertain the status of target system components which may be employed in managing those conditions.
- *plan determination*: An intelligent agent must determine one or more (possibly several alternative) plan(s) for managing the current situation.
- *plan evaluation*: An intelligent agent must evaluate these alternatives to select the 'best' one. In some domains, this may involve simulating the alternatives in order to ascertain their outcomes.
- *plan execution*: An intelligent agent must execute the chosen plan.

While monitoring and execution are relatively straightforward operations, plan determination and plan evaluation may be accomplished in a number of ways that vary in their relative depth of reasoning (Klein & Finin 1987b). In the following sections, we examine a few architectures which distribute these tasks among the human operator and the intelligent system in different ways. The AMVM plays the role of the plan evaluation component in each of these architectures.

2.2 Architectural context: AMVM as top-level model

We may employ the AMVM of section 1 as a 'top-level' model for plan evaluation. Under this scenario, monitoring, plan determination, and plan execution are performed by a human, outside the system. The operator would monitor the target system, identify a set of alternative plans that are applicable in the current situation, and look to the intelligent system to help him choose the best one. The operator would then execute the chosen plan.

The architecture of such a system would incorporate the AMVM as machinery for choosing among competing alternatives. The alternatives themselves would be represented much as in section 1.

2.3 Architectural context: AMVM and a shallow model

Another option is to employ the AMVM as a model for plan evaluation in concert with implemented methods for monitoring, plan determination, and plan execution. We consider two versions of this option, broadly characterized as *shallow* and *deep*.[33]

---

[32] For a discussion of the operational and representational implications of the various options see (Klein & Finin 1987a).

[33] See, e.g., (Hart 1982), (Chandrasekaran 1983), and (Fink 1985) for a discussion of the relative merits of deep and shallow models. See (Klein & Finin 1987b) for an analysis of 'knowledge depth' in the domains of intelligent process control.

By shallow models we usually mean that conclusions are drawn directly from observed facts that characterize a situation. An advantage of shallow models is that they directly encode the heuristics that experts use in performing their reasoning tasks, and are thus relatively easy to build. In addition, shallow models tend to be relatively efficient because they select rather than construct their solutions. One disadvantage of shallow models, however, is that explicitly stating all the preconditions under which a solution should be selected is an error prone process (Koton 1985). Another weakness of shallow models is that they are inflexible, unable to deal with circumstances even slightly different from those explicitly anticipated (de Kleer & Brown 1984). In addition, shallow models may be difficult to maintain, since what is conceptually a single piece of knowledge may be unsystematically distributed across several objects in a knowledge base. Finally, explanations generated from shallow models tend to be limited to traces of the chains of inference that lead to conclusions.

In this section, we consider the integration of the AMVM with a shallow model of plan determination much like that of JESQ. The proposed architecture implements event/response pairs which are chosen by the AMVM according to the attribute values associated with the response portions. The event portions of the pairs, consisting of descriptions of relationships between target system state variable values, describe the eligibility of the plans recorded in the response portions. Each pair represents an operational heuristic, as in JESQ. For example, consider the pairs of Figures 3a and 3b.

```
ALTERNATIVE copy:
    EVENT:
       queue space is low
       dataset
             name = ds1
             size = 1,000,000 lines
             destination = printer1
       printer status
             name = printer1
             status = broken
       tape drive status
             name = tape1
             status = free
    RESPONSE:
       copy ds1 to tape1
       move tape on tape1 to tape library
       move tape from tape library to tape1
       copy ds1 from tape1 to queue
       print ds1 on printer1
       move ds1 from printer1 to user bins
    RESPONSE ATTRIBUTES:
       excess operator time   = 10.0
       excess turnaround time = 34.2
       difference in form     = 1.0
       excess cost            = 1.0
       queue clearing time    = 15.1
END ALTERNATIVE
```

Figure 3a: Alternative 'copy'

```
ALTERNATIVE expensive printing:
    EVENT:
       queue space is low
       dataset
             name = ds1
             size = 1,000,000 lines
             destination = printer1
       printer status
             name = printer1
             status = broken
       printer status
             name = printer2
             status = free
    RESPONSE:
       route ds1 from printer1 to printer2
       print ds1 on printer2
       move ds1 from printer2 to user bins
    RESPONSE ATTRIBUTES:
       excess operator time   = 0.1
       excess turnaround time = 0.0
       difference in form     = 1.0
       excess cost            = 100.00
       queue clearing time    = 25.0
END ALTERNATIVE
```

Figure 3b: Alternative 'expensive printing'

Either heuristic may be appropriate, depending on the status of the target system at the time of instantiation. If say, printer2 is not free but tape1 is, **copy** would be executed. Alternatively, if printer2 is free but tape1 is not, **expensive printing** would be employed. If neither device is free we are out of options.[34] But what if both tape1 and printer2 are free? Which plan is better? This is where our sample AMVM comes into play; we essentially need to choose between the two heuristics according to the installation's operational policy. Recall from section 1 that **expensive-printing** was preferred to **copy**.

In order to implement these ideas and supporting mechanisms, it would be most convenient to use a language which provides for the encoding of both declarative and procedural constructs (e.g., YES/L1 (Cruise et al. 1987), OPS83 (Forgy 1984), YES/OPS (Schor et al. 1986)). A pseudocode sketch of such an implementation appears in Figures 4a-e. In this implementation, procedural code provides a convenient representation for algorithmic knowledge such as coordinating supporting tasks with knowledge-based action.[35] Forward-chaining production systems provide a convenient programming paradigm for expressing data-driven events, including knowledge-based action (plan determination and evaluation) and supporting actions (monitoring). Plan determination is accomplished by the pattern matcher, which invokes operational heuristics whenever they are eligible. The AMVM, an *explicit* model of choice, serves as the conflict resolution algorithm for resolving across them. Recall from chapter II that JESQ employed an implicit model of choice for conflict resolution, and that this gave rise to several difficulties. Monitoring is also coordinated by the pattern matcher, but conflict resolution is arbitrary, since the choice of, for example, which query

[34] Although in an actual implementation, we would also encode heuristics which involve deliberate action to free them.

[35] Recall from chapter II that this was accomplished using priorities in JESQ. Clearly, representing procedural knowledge with procedural code is more natural than hacking at declarative code to make it behave procedurally.

to submit first or which response to acknowledge first is more or less irrelevant. Recall that we labelled arbitrary choosing as an *implicit* model of choice.

```
Procedure JESQ;
begin;
  call initialize;
  Do forever
     begin;
        call monitor;
        call act
     end;
end JESQ;
```

Figure 4a: Shallow model supervisor

```
Procedure monitor;
begin;
   call submit-queries;
   call collect-responses
end monitor;
```

Figure 4b: Monitor

```
Production system act;
conflict resolution = AMVM

{event/response operational heuristics}

end act;
```

Figure 4c: Act

```
Production system submit-queries;
conflict resolution = arbitrary

{rules of the form:

IF   token for query of type x is present
THEN submit query of type x
     destroy token
     create a new token of type x in m minutes }

end submit-queries;
```

Figure 4d: Submit queries

```
Production system collect-responses;
conflict resolution = arbitrary

{rules of the form:

IF   token for response of type x is present
THEN update target system status model
     destroy response }

end collect-responses;
```

Figure 4e: Collect responses

This pseudoprogram exemplifies:

1. how the AMVM may be incorporated into the production systems architecture to implement knowledge-based conflict resolution,
2. how implicit and explicit models of choice can coexist according to the requirements of the task at hand,
3. how declarative and procedural representations may be integrated to promote naturalness of expression, and
4. how a JESQ-like system might be enhanced.

Elaborating on (4), note that this skeletal architecture might be sufficiently general to accommodate other intelligent process control domains as well. It might even be the case that this architecture could serve as a generally useful extension to the production systems architecture, with applications beyond intelligent process control.

The standard production systems architecture may be depicted as in Figure 5.

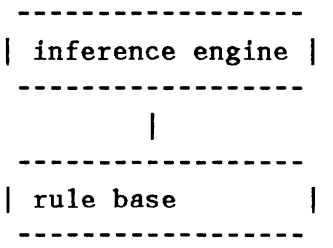

Figure 5: Standard production systems architecture

The extension may be depicted as in Figure 6. The choice model database may contain any number of AMVM's, code for arbitrary conflict resolution, and implicit hard-wired conflict resolution algorithms.[36] Rules are organized in blocks according to their logical function, and programmers explicitly name the model of conflict resolution to be used for each particular block.

[36] This is to be distinguished from previous efforts that employ utility functions for conflict resolution in rule-based systems (e.g., White & Sykes 1986), which assume a single explicit model of choice.

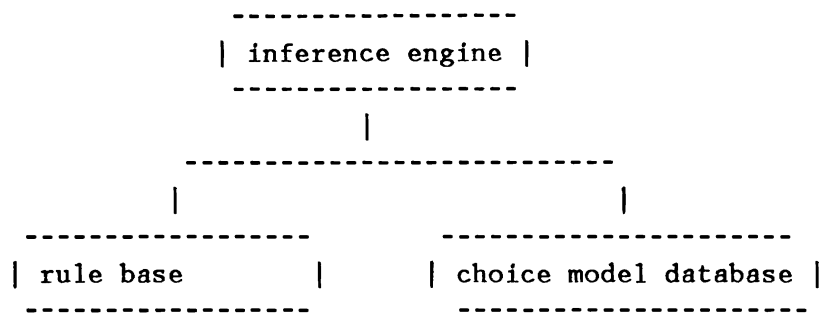

Figure 6: Augmented production systems architecture

The augmented production systems architecture may in turn be naturally integrated with procedural code in the spirit of YES/L1-like languages (Cruise et al. 1987) as in Figure 7. In this architecture, production rules coexist with procedural code as in our pseudocode formulation of JESQ. The supervisory interpreter directs the execution of procedures and production systems as specified, employing the choice model database for conflict resolution as in Figure 6.

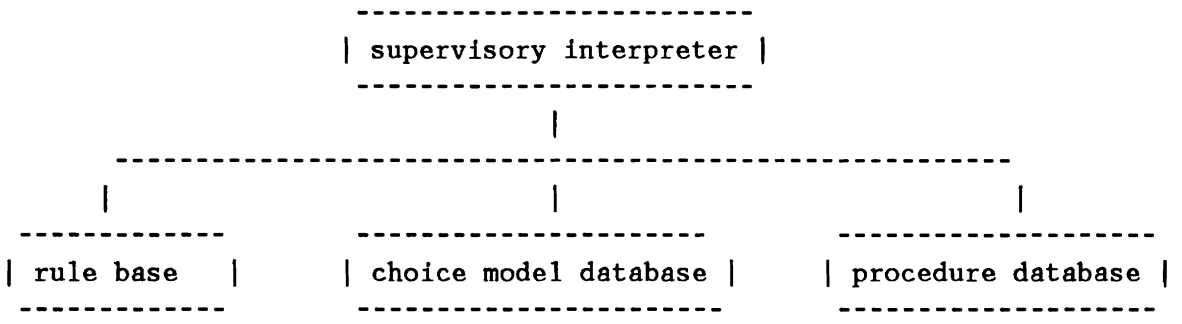

Figure 7: Integrated augmented production systems and procedural architecture

In summary, we can envision use of the AMVM in shallow intelligent systems where explicit, knowledge-intensive choices are necessary. An arbitrary number of AMVM's may be encoded (each pertaining to a given production system module), and these may coexist with implicit models of choice.

2.4 Architectural context: AMVM and a deep model

The plans encoded in the shallow model of the previous section, which specify the movement of data from the queue to other components (e.g., tape drives, printers), are based on the structure of the underlying computer system being modelled. But since this structure is only implicitly represented, the system would be limited by some of the disadvantages which typify shallow models. For example, the configuration of the computer system may be changed, requiring additional preconditions to be encoded in the antecedents of rules, but there is no systematic way of identifying the rules that should be changed. Depending on the nature of a particular configuration change, we may also need to add or delete rules. Another potential limitation of the system is that explanations are limited to the presentation of the conditions under which plans are applicable. Finally, the system would only be able to handle those state conditions that have been encoded in the antecedents of rules, with no provisions for dealing with unanticipated situations.

In contrast, deep models of expertise correspond more closely to the notion of reasoning from first principles. They tend to be more robust than shallow models, handling problems not explicitly anticipated and exhibiting higher performance at the periphery of their knowledge. In addition, it can be easier to verify the completeness of deep models. For example, in device-centered models of physical systems (e.g., de Kleer & Brown 1984, Davis 1984) each physical device maps directly into a structured object in the representation. Deep models of expertise are also more useful for generating explanations in that reasoning steps which are usually implicit in shallow models can be elucidated. Deep reasoning is, however, bound to be slower and more complex than shallow reasoning in that a more sophisticated control structure is required (Koton 1985).

In this section, we describe a deeper model for managing queue space. Monitoring, plan determination, plan evaluation, and plan execution are performed automatically as in the shallow model. But plan determination is a process of *generation* rather than *invocation*, which employs an explicit structural representation of JESQ's domain as depicted in Figure 8.

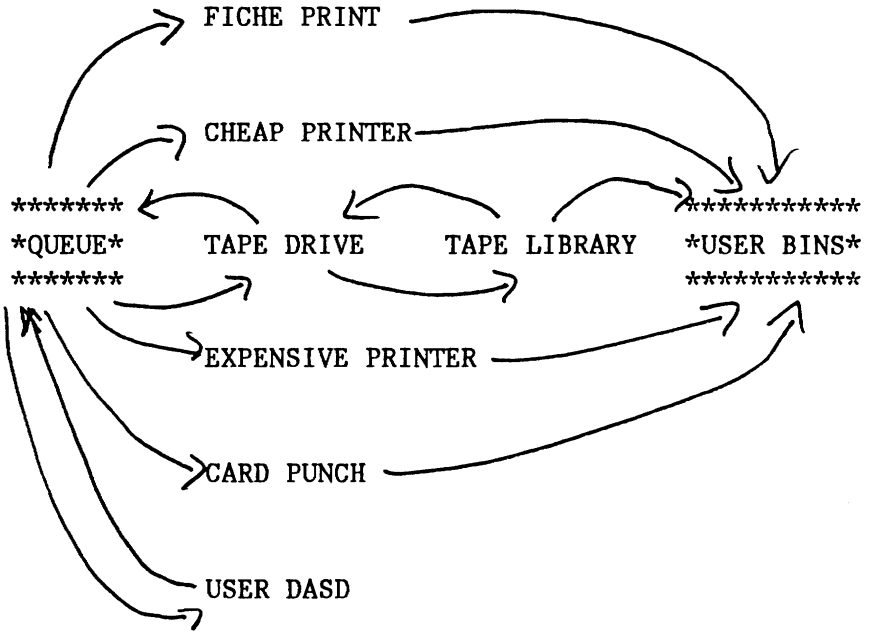

Figure 8: Structural model of computer installation

Using this model, plan determination adopts the form of searching a graph in which nodes represent components and edges represent their interconnections. The search always begins at node QUEUE and terminates at node USER BINS, and each such path through the graph (allowing one iteration of any given cycle) represents a candidate path through which datasets may flow in order to clear the QUEUE. This representation has the advantage that a change in the configuration may be directly mapped into a change in the representation, and the knowledge about computing plans for moving datasets (searching the graph) remains unchanged. It is also more portable than the other representations, requiring only a configuration description for any particular installation.

As for the other component tasks, monitoring encompasses supplying values for status variables associated with each node in the graph (e.g., the availability of a tape drive). Plan evaluation involves using the AMVM to compare alternative generated paths. For each component of Figure 8, we encode a description (e.g., processing time per line of data) which supports the computation of the attributes of the AMVM. A plan (path of devices) can be evaluated with respect to the attributes (e.g., turnaround time) by summing the values associated with the processors that lie along the generated path. One advantage of this method is that we need only supply local device-dependent data for each represented device in order to compute the attributes associated with generated plans, rather than assigning them at the level of composite plans as in the two architectures

previously discussed. Execution encompasses executing commands associated with each interconnection to move the dataset from one device (node) to another.

2.5 Interfacing plan determination with evaluation

We have outlined three architectures in which the AMVM served as the machinery for plan evaluation. Let us make explicit the common interface with plan determination shared by each of these. In the first architecture discussed, the operator specified the alternatives and the attributes. In the second, these alternatives were hard-coded in the consequents of rules, as were the associated attribute values. In the third, the plans were generated rather than invoked, and the corresponding attribute values were computed from the same explicit model used for plan determination. It is important to note that in all three architectures, we can interface plan determination with evaluation in the same way: by supplying the values for attributes associated with alternative plans (be they directly captured from users, encoded ahead of time, or computed in realtime).

2.6 Implications for approach to explanation and refinement

Given this common interface, it seems reasonable to first focus on the development of explanation and refinement facilities in isolation. This is not to say that either the explanation or refinement of choice models is totally isolated from the explanation or refinement of other structures (e.g., rules in the second architecture or device descriptions in the third). Indeed, integrating the explanation and capture of choice knowledge with that of other knowledge is an interesting and formidable task. The point is that the existence of a uniform interface to the AMVM provides reason to believe that machinery for the explanation and refinement of choice knowledge can be integrated with analogous machinery for supporting other structures at some future time. Thus, we focus on the explanation and refinement of choice-related information first, with an eye toward facilities which support more complex architectures encompassing other structures (e.g., for plan invocation or generation).

3. Acquiring new alternatives

Of the three architectures discussed, only the third provides for the automated generation of alternatives. The first two require that the alternatives be specified. Before addressing explanation and refinement in general terms, we wish to demonstrate the ease with which new alternatives may be integrated with existing ones through the use of the AMVM. Recall from chapter II that the lack of such a model in JESQ rendered the rational integration of new alternatives nearly impossible.

Suppose that an installation acquires a new low-end printer. This creates a new alternative for output processing at the site.[37] We envision the following sort of dialog between an intelligent system and an operations manager for capturing the information needed to integrate this alternative into the knowledge base. The user initiates the dialog; his participation is shown in CAPITALS. Intelligent system-generated text is shown in normal case. The mechanisms underlying this text are discussed in section 6.

> NEW ALTERNATIVE

Please type the name of the alternative.

> CHEAP PRINTING

[37] For this example, assume that cheap printing is not yet included in the set of alternatives.

How many minutes of additional operator time for cheap printing? As a reference point, additional operator time for dasd is .1 minutes, and is 180 minutes for install.

> .1

How much additional turnaround time (in minutes) for cheap printing? As a reference point, additional turnaround time for dasd is 32.1 minutes, and is 180 minutes for install.

> 10

How much difference in form (on a scale from 0 to 10, with 0 the most different) for cheap printing? As a reference point, difference in form for dasd is 10, and is 10 for install.

> 8

How much additional material cost (in dollars) for cheap printing? As a reference point, additional material cost for dasd is .50 dollars, and is 5000 dollars for install.

> 0

How much queue clearing time (in minutes) for cheap printing? As a reference point, queue clearing time for dasd is 1 minutes, and is 210 minutes for install.

> 40

Assuming queue space is, say, just a little low, the new ordering on alternatives is:

dasd, expensive-printing, cheap-printing, copy, delete, cards, fiche, install

Does that seem reasonable?

> YES

Good.

We expect that in general, the addition of alternatives will remain this simple. Contrast this with attempting to guess the correct priority for the newly-added alternative.

Had the ranking been incorrect, the user would invoke the explanation and refinement facilities described in the following sections,[38] for the model of choice itself would have to be modified.

4. Justifying choices

The purpose of explaining the rationale behind choices is twofold. First, we need to convince the user that the chosen alternative is indeed the preferred one. Second, we need to provide insight into how the current choice model behaves so that erroneous or dated portions of it can be identified for modification.

We communicate the flavor of UTIL's proposed explanation facility as follows. Section 1 describes the elements of justifications collected from computer operators. Section 2 describes a potential approach to explanation and associated commands. Section 3 depicts a hypothetical dialog between

[38] Section 6 contains an example of refinement following the incorrect integration of a new alternative.

UTIL and the user. Some potential mechanisms for supporting this dialog are presented in section 4. Section 5 reflects upon the integration of these mechanisms with other explanation facilities.

4.1 Some discourse elements

Our goal is to generate convincing justifications for choices. But what makes a justification convincing? How do people explain choices to one another? In an effort to answer these questions, we conducted a series of interviews with twelve computer operators at IBM's Thomas J. Watson Research Center in Yorktown Heights, N.Y. Each operator was presented with the following description of a choice situation concerning the management of JES queue space:

Queue space is low. We have a large dataset destined for a 3800 printer, to be printed on 1part forms, but the 3800 is down and awaiting repairs. The following alternatives are available:

- **DJ to tape (DJ)**[39]
- **print on a 3211 (3211)**[40]
- **call the user and negotiate the fate of the dataset (CALL)**
- **do nothing; wait until the 3800 has been fixed (WAIT)**
- **install a new printer just to print the waiting dataset (INSTALL)**

The operator was then asked to rank the actions. Next, the operator was asked to compare various alternatives according to the ranking. Two sorts of comparisons were requested. First, we asked why the chosen alternative was the best one. For example, if the operator produced the ranking DJ > CALL > 3211 > WAIT > INSTALL, we asked, 'Why is DJ the best?' Next, we asked the operators to compare particular pairs of alternatives, as in 'Why is DJ better than CALL?' and 'Why is CALL better than 3211'? Observations regarding both the substance and form of their justifications are discussed in the following sections. Where portions of a justification have been reproduced, we have omitted rambling, 'thinking out loud', and pause words such as 'ugh' and 'hmm'. The essential structure and content, however, have been preserved.

Focus on values and objectives

Our interviews provide another data point to support Keeney's assertion that 'values are the basis for any interest in any decision problem' (Keeney 1986). When asked to compare alternatives, operators refer to the objectives, their associated attributes, levels of attainment for particular alternatives, and other value-oriented objects in justifying their choices.

We observe that the operators mention both high- and low-level objectives in justifying their choices. Justifications sometimes include a statement of general mission, as in:

'As operations, our responsibility is to have the system up and as much as possible operational at all times.'

'That's what we're here for, to *service the users*'

'We need the *best and quickest solution* we can find to recover the error. *That's what operations is all about.*'

'We're *serving the people*, you know.'

---

[39] This is operator jargon for alternative **copy**.
[40] This is alternative **cheap printing**.

The operators are sometimes more specific about which objectives are best satisfied by alternatives, as in:

'... by the time it prints out or whatever, that's a *slower* process.'

'It saves the operator a lot of *work*'

'well, its the *quickest* way and the *easiest* way'

These utterances, for example, refer to objectives such as *minimize additional operator work* and *minimize queue clearing time.*

Pruning and ordering objectives

In justifying a choice, the operators focus on those objectives which particularly distinguish the alternatives under consideration. If one objective value greatly dominates the decision, for instance, other objectives may be omitted from the justification. For example, when asked why WAIT was preferred to INSTALL, one operator just laughed and proclaimed, 'cost!'. Yet that same operator, in justifying why DJ was better than WAIT, replied, '... going out to tape is quick'. While both cost and speed clearly influenced his decisions, he focussed on the objective which truly distinguished the pair of alternatives under consideration in each explanation. A corollary to this observation is that objectives that are not pruned from the explanation should be ordered in terms of their impact in distinguishing the two alternatives.

Granularity of attribute values

In general, the operators mention qualitative values for attributes, as in:

'That's an *enormous* expense'

'I'm assuming that the queue space is *real full* here'

'It saves the operator *a lot* of work'

However, we also note occurrences of quantitative values in the context of more detailed explanations, particularly in discussing particular plans as in:

'You can contact the user in say *10 seconds.* You say he's got this dataset and he's got to look at it. You can solve your problem or make your decision in like *a minute*'

Thus, it seems that high-level justifications encompass qualitative values while more detailed justifications for particular plans refer to quantitative estimates of those values.

Stating the importance of objectives

We observe that operators make explicit statements regarding not only the objectives that underlie choices, but also the importance of those objectives. Consider, for example:

'An operator's time is important'

'Time in operations is very important'

These operators felt compelled to mention not only that minimizing an operator's time was an objective, but that it was generally important, outside consideration of any particular alternatives.

Explicitly identifying the decision making context

We observe that operators not only identify the elements of decision making (the objectives, their relative importance, etc.), but also the *context* in which the relationships between those elements are assumed to hold. For example:

'If the system's going down or something, you go with the quickest'

'If queue space is at an intermediate point where there's not going to be much problem, I'd just let it sit there.'

'I'm not too crazy about destroying the person's dataset, but *it all depends on how critical the situation is'*.

'I'm assuming that the queue space is real full here'

These utterances imply that different alternatives would be chosen under different circumstances. In the queue space domain, these 'circumstances' correspond to the severity of the current situation as measured by the amount of space left on the JES queue.

Referring to significant components of plans

Values for attributes which describe a plan may represent sums of more detailed attribute values contributed by individual plan steps. For example, the increase in turnaround time due to DJ represents the sum of component increases in turnaround time yielded by the individual actions of the DJ plan, including mounting tapes, labelling them, submitting the 'copy command', and other detailed actions.

We observe that the operators do not merely present the values of (summary) plan attributes, but additionally identify the most significant component plan steps, as in:

'They'd have to get the tapes out, label them, you know'

'You can contact the user in say 10 seconds. You say he's got this dataset and he's got to look at it. You can solve your problem or make your decision in like a minute'

Thus, operators mention the values for significant attributes which characterize plans, and expose which components of those plans most significantly account for those values.

Defending alternatives by identifying others' weaknesses

We observe that the operators often extoll the virtues of an alternative by pointing out the comparative weaknesses of others. For example, consider:

'Take one of the other ones. By the time you get the tape or whatever, by the time it prints out or whatever, that's a slower process.'

'Maybe the guy can't use it when it comes out on a 3211'

These utterances focus on the negative consequences of alternatives with regard to 'maximize queue clearing time' and 'minimize difference in form the user receives'.

Reference to other decisions and decision makers

We observe that the operators make references to the decisions of other agents and decision makers. For example, consider:

'Some people here might think that their waiting time is a lot more valuable than the cost of the new printer, but that's their opinion'

'That's fine as long as we have authorization from high above. Then we handle it differently'.

' ... We can do it without asking about the decision we have to make.'

'Based on our own experience or the title you have, you may make your own decision.'

'*On my own*, chances are that I would not delete the dataset'

Summary and continuing work

We have identified some of the essential elements of the operators' justifications for their choices, and we will continue to examine the dialogs for additional elements.

In addition, it seems logical to collect dialogs from decision analysts as well. While we believe that the actual domain experts' (i.e., operators') justifications are probably the better model for designing UTIL's explanation facility, observing trained decision analysts as they justify choices to their clients should yield additional useful elements of explanation.

4.2 Approach to explanation and associated commands

While there may be many possible approaches to elucidating the factors underlying a choice (e.g., graphical or tabular presentation of the underlying attributes), our goal is to produce natural explanations akin to those which people seem to offer in justifying choices. However, the reader should note that the current enterprise is *not* a study in discourse analysis or in natural language generation; we are here concerned with incorporating elements of both the form and content of justifications for choices as people present them, but are not focussed on precisely reproducing human-like justifications. There are three reasons for this:

1. Explanation, in part, serves as the user's 'window' into the structure of the AMVM. Thus, we are faced with the challenge of providing adequate insight into the model's parameters and operation, while providing intuitively appealing justifications for choices.
2. People use pause words, are needlessly verbose, and commit other linguistic acts which potentially obscure justifications. We want to avoid replicating these characteristics of human explanations.
3. Natural language generation is a challenging research area in its own right. It would be impractical to attack problems in that field in the context of this research, the goals of which are quite different. Following researchers such as Davis (1976) and Langlotz et al. (1986), we will use simple methods for text generation so as not to deviate from our research focus. Our hope is that our efforts may be adapted to employ more sophisticated methods for text generation at a later time.

These considerations give rise to the following design strategy for UTIL's explanation component: borrow the seemingly desirable characteristics of both the form and substance of human justifications for choices (not being distracted by grammatical concerns such as tense agreement and proper pluralization), and embellish these with additional information which seem to allow for effective model repair. In short, we need to strike a balance between naturalness, model exposition, and research focus in generating explanations.

In this section we describe some of the explanation-oriented commands which we would like to offer users in UTIL. Naturally, all of these involve the comparison of alternatives in one form or another. We assume that AMVM-based evaluation of alternatives is initiated via a command of the form **(CHOOSE** $a_1, \ldots, a_n$ **)** where $\{a_1, \ldots, a_n\}$ is a set of alternatives from which CHOOSE returns the best. In the top-level architecture of section 2.2, the user issues CHOOSE directly. In the shallow model of 2.3, $a_1, \ldots, a_n$ would be instantiated by the pattern matcher, and the interpreter would invoke CHOOSE as the conflict resolution step of the recognize/act cycle. In the deep model of section 2.4, a controlling module would invoke CHOOSE as the plan evaluation phase of the monitor/plan determination/plan evaluation/execution cycle.

First, we describe the commands which the user would use immediately after invoking CHOOSE. **(WHY)** allows the user to ask why the preferred alternative is the best, without specific reference to another. Looking to human interaction for clues, consider the following operator's response to 'WHY is CALL the best?':

Well, its the quickest way and easiest way. You can contact the user in, say, 10 seconds. You say he's got this dataset and he's got to look at it. You can solve the problem or make your decision in like a minute. Take one of the other ones, by the time you get the tape or whatever, by the time it prints out or whatever, that's a slower process. You know, you look at the easiest way and go from there, or quickest way. If the system's going down or something, you go with the quickest.

This operator compared his choice, CALL, with his next two choices, DJ and 3211. In addition, he described the significant steps in plan CALL which made it desirable with respect to objective 'maximize speed'. He also explicitly indicated that 'maximize speed' was a generally important objective in the decision making context 'system going down'.

A more focussed version of WHY is **(WHY NOT b)**, which allows the user to compare the chosen alternative with a particular alternative **b** that he might think more appropriate. WHY NOT should provide more specific responses than WHY, focussing only on the two mentioned alternatives, and narrowing in on precisely those objectives which serve to distinguish their relative desirability. We speculate that WHY NOT would probably be repetitively used after invoking WHY; that is, WHY will provide a general justification for the chosen alternative which the user will challenge via use of WHY NOT.

The operation of WHY and WHY NOT reflects the assumption that CHOOSE has just been executed. We should also supply some commands which provide additional information by calling CHOOSE and using its data structures or intermediate results. For example, the user may gain additional insight into the AMVM's behavior by comparing two alternatives, neither of which was returned by CHOOSE. **(COMPARE a b)** provides a WHY NOT analysis for alternatives **a** and **b** (after EVALing (CHOOSE a b)). We envision that many such simple variations will obviously present themselves as we implement UTIL.

It is also important to allow the user to focus on details of a justification that interest him and to ask for the more general context of a particular part of the justification. These operations naturally correspond to the descending and ascending of the objectives hierarchy, respectively. In order to obtain more detail regarding an objective, we might provide a **(DETAIL o)** command, where **o** is an objective (perhaps mentioned in an explanation generated by WHY or WHY NOT). DETAIL

may be used many times in succession to elicit progressively more detailed explanations. DETAIL may also be used outside the scope of a particular choice (i.e., when CHOOSE has not been executed), to elucidate portions of the model.

So that users may ascertain the context of a detailed objective, we might provide a **(RELEVANCE o)** command, which elucidates the parent and sibling objectives for objective **o**. We can also envision the potential usefulness of an **(EXPLAIN o)** command which provides the information of both DETAIL and RELEVANCE for an objective **o**.

We might also provide an alternative interface whereby commands such as EXPLAIN, RELEVANCE, and DETAIL are entered by highlighting a portion of a previously generated explanation on the screen and clicking a mouse or hitting a PF key.

4.3 Hypothetical dialog

Having provided a brief description of potential UTIL commands for explanation, we present a hypothetical dialog and hint at potential implementation methods. A disclaimer: we are not ready to advocate these methods; we will need to implement them and to test them on several examples. They are presented here only to communicate the flavor of the approach we plan to pursue at the outset of the research.

So as not to obscure its naturalness, we present the dialog in its entirety in this section, and defer description of the hypothesized underlying mechanisms to the next section. The syntax of the dialog is as follows. User responses follow the prompt symbol (>) and are CAPITALIZED. Text generated by UTIL appears in ordinary case. Annotations are {enclosed in braces}. Assume that UTIL has access to all the information in the sample AMVM of section 1.

> CHOOSE COPY, EXPENSIVE PRINTING, CHEAP PRINTING, DELETE, INSTALL, FICHE, CARDS, DASD

My recommendation is to perform plan dasd.

> WHY

My objective is to effectively manage queue space. In order to achieve that objective I must maximize user satisfaction, minimize material costs, minimize queue clearing time, and minimize time spent by the operator. This is similar to the way in which I choose among actions to recover from hardware problems.

Assuming that queue space is a little low, dasd is better than expensive printing, for example, primarily because it is of lower material cost and lower clearing time. expensive printing is of higher extra material cost because it involves using $100 worth of paper.

dasd is better than cheap printing primarily because it is of lower clearing time. cheap printing is of higher clearing time because it involves using a slow printer.

By the way, dasd would also be the choice of most 1st shift managers and 2nd shift managers.

> WHY NOT CARDS

Although cards is slightly better than dasd with regard to additional turnaround time (a component of user satisfaction), dasd is much better with regard to the difference in the form the user receives (a component of user satisfaction).

{The user enters DETAIL USER SATISFACTION or highlights the phrase 'user satisfaction' and clicks the DETAIL mouse button}

My ability to maximize user satisfaction depends upon the degree to which I minimize additional turnaround time and minimize the difference of the form the user receives. additional turnaround time and difference in form are equally important in my evaluation of user satisfaction.

While cards is characterized by additional turnaround time of 15 minutes, less than 32.1 minutes for dasd, dasd is characterized by no difference in the form the user receives, much more similar to the user's request than cards. Since additional turnaround time and difference in form are equally important in my evaluation of user satisfaction, dasd is clearly better with regard to user satisfaction.

{The user enters DETAIL ADDITIONAL TURNAROUND TIME or highlights this phrase and clicks the DETAIL mouse button}

additional turnaround time is the amount of additional time the plan delays the user's acquisition of his output, measured in minutes. The less additional turnaround time, the better.

> RELEVANCE ADDITIONAL TURNAROUND TIME

additional turnaround time is a component of user satisfaction, along with difference in form. user satisfaction is a component of effectively manage queue space.

> DETAIL CHOICE

My objective is to effectively manage queue space. In order to achieve that objective I must maximize user satisfaction, minimize material costs, minimize queue clearing time, and minimize time spent by the operator. Although cards is slightly better than dasd with regard to additional turnaround time (a component of user satisfaction), dasd is much better with regard to the difference in the form the user receives (a component of user satisfaction), somewhat better with regard to clearing time, and slightly better with regard to additional material cost.

> EXPLAIN CLEARING TIME

clearing time is the amount of time the plan takes to clear a dataset off the queue, measured in minutes. The less clearing time, the better. clearing time is a component of effectively manage queue space.

> WHY NOT INSTALL

dasd is as good or better than install with regard to every objective which underlies my choice.

{And the user continues to explore the rationale behind the decision as desired}

4.4 Mechanisms employed in the hypothetical dialog

In this section we provide a description of some of the mechanisms which might be implemented to produce the dialog of the previous section.

All responses are generated by instantiating values in response templates. The templates themselves would be selected on the basis of the context of the dialog and of the nature of the explanation appropriate to the situation.

The syntax of the dialog is as follows. As before, user responses follow the prompt symbol (>) and are CAPITALIZED, and text generated by UTIL appears in ordinary case. Variables in response templates are < enclosed in angle brackets > . All elements of dialog are in **bold**, and descriptions of methods used to generate UTIL's responses appear in normal font. Assume that UTIL has access to all the information in the sample AMVM of section 1.

> CHOOSE COPY, EXPENSIVE PRINTING, CHEAP PRINTING, DELETE, INSTALL, FICHE, CARDS, DASD

My recommendation is to perform plan < dasd > .

AMVM-based evaluations (Table 3) indicate that dasd is the highest valued option.

> WHY

My objective is to < effectively manage queue space > . In order to achieve that objective I must < maximize user satisfaction > , < minimize material costs > , < minimize queue clearing time > , and < minimize time spent by the operator > .

UTIL examines the objectives hierarchy and lists the top-level objectives in order of absolute importance as recorded in the value function. Recall that our value function is:

$$v(x1,x2,x3,x4,x5) = .1*v1(x1) + .5(.5*v2(x2) + .5*v3(x3)) + .2*v4(x4) + .2*v5(x5)$$

Note that detailed objectives (turnaround time and difference in form) are suppressed.

This is similar to the way in which I choose among actions to < recover from hardware problems > .

A *user profile* (UP) is maintained for each UTIL user, consisting of a set of value functions and associated objectives hierarchies which the user has constructed, each indexed by the name of the choice to which it pertains and the context description under which it is applicable.

To make analogies with other choices the user has faced, we search the user's UP for a model that includes the same objective hierarchy under the same context, in this case presumably matching on a choice called 'recover from hardware problems'. The objective weights in the value function need not be the same for a successful match.

Assuming that queue space is < a little low > ,

Context identification is performed simply by enumerating context variables. For choice 'effectively manage queue space', there is a single context variable 'amount of space left'. The description 'a little low' might correspond to the range 20-40% left. Another context, 'extremely low' might correspond to 10-19% left. Each context is associated with a value function which differs from others in its weights on objectives.

< dasd > is better than < expensive printing > for example, primarily because it is of < lower > < material cost > and < lower > < clearing time > .

Our value function induces the ordering: dasd > expensive printing > cheap printing > copy > delete > cards > fiche > install. Looking to the human operators' explanations for queues, UTIL explains why dasd is better than its two closest contenders, in this case expensive printing and cheap printing. In response to WHY, we focus only on positive evidence, since WHY is interpreted as requesting a general argument in support of the chosen alternative. Detailed analysis of the actual tradeoffs involved in choosing one alternative over another is handled by WHY NOT.

We employ the following method for enumerating the 'key' objectives underlying the choice over close contenders in response to WHY. Note that a different method is used for WHY NOT, which produces more focussed explanations and addresses tradeoffs.

1. Determine if the chosen alternative *dominates*[41] the contender by direct comparison of attributes values. If so, simply explain that the chosen alternative is as good or better with regard to all the objectives which underlie the decision. If not (the usual case), continue.
2. Calculate the differences in *contribution* ($w_i v_i(x_i)$) of each attribute i with respect to the two alternatives. The contribution for an attribute yields a measure of the effects of that attribute on the overall evaluation. This measure takes into account both the value and the importance of the attribute with respect to a particular alternative.[42]
3. Prune attributes which produce negative or zero differences. This leaves only those attributes which support the chosen alternative.
4. Starting with the attributes which have the largest contribution differences, collect enough evidence in support of the alternative to counterbalance the total of the negative evidence. This, in a sense, yields the set of 'important' attributes which support the chosen alternative, for this set has the following properties: (i) It includes those attributes which provide the largest positive contribution differences and (ii) It includes enough of these to counterbalance the attributes which argue against the chosen alternative. Loosely speaking, attributes providing positive contribution differences which are omitted are merely the 'icing on the cake'.
5. Present the resulting set of attributes in defense of the alternative, along with the higher-level objectives influenced (if any exist).

Following is a detailed exposition of the method for this example.

Step1: We compare the raw attribute levels of the two contenders (+ is better, - is worse, = is equal):

dasd:	.1	32.1	1	.5	1
expensive-printing:	.1	0	1	100	25
	=	-	=	+	+

We observe that dasd does better on cost and clearing time, but worse on turnaround time. So dasd does not dominate expensive printing. Rather, it was chosen because the value function specifies a tradeoff of turnaround time for better cost and clearing time.

Step2: To identify the most important objectives which distinguished the alternatives, we calculate the differences in the contributions made by the individual attributes:

dasd:	.1	.1	.25	.2	.2
expensive-printing:	.1	.25	. 25	.08	.12
differences:	0	-.15	0	+.12	+.08

[41] **Definition**: Suppose, without loss of generality, that preferences increase in each x_i (i.e., the more x_i, the better) for a set of attributes $x_1, \ldots, x_n$. We say that an alternative $(x_1, \ldots, x_n)$ *dominates* another alternative $(y_1, \ldots, y_n)$ whenever $x_i \geq y_i$ for all i and $x_i > y_i$ for some i. Less formally, if one alternative dominates another, then the dominated alternative is a 'noncontender' for best, since the dominating alternative is at least as good for every attribute and strictly better for at least one. In the usual case, where neither alternative dominates, the decision-maker's tradeoffs between attributes come into play.

[42] To compare the numbers returned by different component value functions (corresponding to different attributes) for a given alternative as well as the values returned by the (composite) value function across alternatives, we need to assume that v is a *measurable value function*. See (Dyer & Sarin 1979) for details.

Step3: Pruning negative and zero differences to remove negative and non-distinguishing attributes leaves x4 (contribution difference = .12) and x5 (.08).

Step4: We prune the least important positive difference attributes as follows. Starting with the highest valued positive contribution, we observe whether it is sufficient to cancel the sum of the negative evidence:

$$|.12| > |-.15| ?$$

The answer is 'no' in this case, so we include the next highest valued positive attribute contribution:

$$|.12 + .08| > |-.15| ?$$

The answer is yes, so we are finished.

Step5: x4 and x5 are presented; neither are components of higher level attributes.

< expensive printing > is of < higher > < extra material cost > because it involves < using $100 worth of paper > .

For each plan, we maintain a canned description of the steps which most influence its attribute values. To instantiate 'using $100 worth of paper', we referenced the description for plan 'expensive printing' for attribute 'material cost'. Attribute 'material cost' was selected rather than 'clearing time' because it was associated with the larger contribution difference.

Next, we repeat the process with the second closest contender, < cheap printing > .

< dasd > is better than < cheap printing > primarily because it is of < lower > < clearing time > .

Step1:

dasd:	.1	32.1	1	.5	1
cheap-printing	.1	10	.8	0	40
	=	-	+	-	+

In this case, the existing model specifies a tradeoff of better difference in form and clearing time for worse turnaround time and material cost.

Step2:

dasd:	.1	.1	.25	.2	.2
cheap-printing:	.1	.2	.2	.2	.08
differences:	0	-.1	+.05	0	+.12

Step3: Pruning negative and zero differences, we have x3 and x5 as candidates to mention as positive evidence for selecting dasd.

Step 4: We compare the largest positive difference with the sum of the negative differences.

$$|.12| > |-.1| ?$$

The answer is yes, so we needn't include x3 in the explanation.

Step5: We present x5 in the justification. Had x3 not been excluded, we would have included something like '<lower> <difference in form> helps to <maximize user satisfaction>' in the justification.

<cheap printing> is of <higher> <clearing time> because it involves <using a slow printer>.

To instantiate <using a slow printer>, we reference the description for plan <cheap printing> for attribute <clearing time>.

By the way, <dasd> would also be the choice of most <1st shift managers> and <2nd shift managers>.

In a prior exposition, the user profile was used to organized different choice models (intraprofile comparison) for a given user so that analogies with other problems could be made; here, we use the profiles to compare the preferences of different users (interprofile comparison) for a given problem. Specifically, we examine the user profiles of 1st and 2nd shift managers, each of which includes the 'effectively manage queue space' choice model. UTILI executes these, noting that they produce the same choice as the current user's model for the same context description.

We might also explore the organization of UPs in a *user profile hierarchy* which logically groups decision makers to reflect the hierarchical structure of their organizations. For example, we might group 1st and 2nd shift managers under the category 'managers', and use this structure to make statements about managers in general. This is somewhat reminiscent of the *stereotype DAG* of Rich (1979).

> WHY NOT CARDS

Although <cards> is <slightly> better than <dasd> with regard to <additional turnaround time> (a component of <user satisfaction>), <dasd> is <much> better with regard to <the difference in the form the user receives> (a component of <user satisfaction>).

The following variation on the WHY procedure might be used. Whereas WHY presents only the most influential evidence which supports the alternative, WHY NOT indicates tradeoffs, exposing both positive and negative evidence underlying the choice of an alternative.

1. Determine if the chosen alternative dominates the contender by direct examination of the attributes. If so, simply explain that the chosen alternative is as good or better with regard to all the objectives which underlie the decision. If not (the usual case), continue; we must expose the tradeoffs that were taken into account.
2. Calculate the differences in contribution for each attribute.
3. Prune least important positive and negative attributes (by an extension of the method for WHY which is illustrated below). This leaves those attributes which most influenced the decision.
4. Present these attributes, along with the higher-level objectives influenced (if any exist).

Following is a detailed exposition of the method for this example.

Step1: We check to see whether dasd dominates cards:

```
dasd:    .1   32.1   1   .5   1
cards:   .1   15    .1   20   45
         ---------------------
          =   -      +   +    +
```

dasd does not dominate cards. Rather, selecting dasd implies a tradeoff of worse turnaround time for better difference in form, material cost, and queue clearing time.

Step2: We compute the contribution differences:

dasd:	.1	.1	.25	.2	.2
cards:	.1	.15	.025	.16	.04
differences:	0	-.05	+.225	+.04	+.16

Step 3: As in WHY, we do horizontal pruning of the objectives hierarchy, but this time accounting for negative influences as well. Should the user desire more information, he can use the DETAIL command.

We proceed as follows. First, we compare the negative and positive differences with the largest absolute values. If the positive differences fail to outweigh the negative differences, we continue to add positive differences until they do. For this example, the largest negative difference is contributed by x2 (-.05). The largest positive difference is due to x3 (+.225). Since |.225| > |-.05|, we are done.

Step 4: We present x2 and x3 to the user. Qualitative differences in utility (e.g., 'much') are assigned according to range of differences in the utilities of the chosen attributes (e.g., .7 to 1). In this example we have:

	x2	x3
dasd:	.4	1
cards:	.6	.1
differences:	-.2	+.9

So dasd is 'much' better with respect to x3, and 'slightly' worse with respect to x2.

> DETAIL USER SATISFACTION

My ability to < maximize user satisfaction > depends upon the degree to which I < minimize additional turnaround time > and < minimize the difference of the form the user receives > . < additional turnaround time > and < difference in the form the user receives > are < equally > important in my evaluation of < user satisfaction > .

This is accomplished by instantiating objective names and weights from the objectives hierarchy.

While < cards > is characterized by < additional turnaround time > of < 15 > < minutes >, < less > than < 32.1 > < minutes > for < dasd >, < dasd > is characterized by < no > < difference in the form the user receives >, < much > < more similar to the user's request > than < cards > . Since < additional turnaround time > and < difference in form > are equally important in my evaluation of < user satisfaction >, < dasd > is clearly better with regard to < user satisfaction > .

The method is essentially the same as previously described, but focussing on user satisfaction and unconditionally mentioning all attributes.

In addition, the weights on objectives are displayed by simple inspection of the objectives hierarchy.

> DETAIL ADDITIONAL TURNAROUND TIME

< additional turnaround time > is < the amount of additional time the plan delays the user's acquisition of his output > , measured in < minutes > . The < less > < additional turnaround time > , the better.

Associated with each attribute is a textual description of the attribute, its scale of measurement, whether more or less is better, and other information. Simple lookup and instantiation is used to generate the explanation.

> RELEVANCE ADDITIONAL TURNAROUND TIME

< additional turnaround time > is a component of < user satisfaction > , along with < difference in the form the user receives > . < user satisfaction > is a component of < effectively manage queue space > .

This is simple traversal of the objectives hierarchy.

> DETAIL CHOICE

My objective is to < effectively manage queue space > . In order to achieve that objective I must < maximize user satisfaction > , < minimize material costs > , < minimize queue clearing time > , and < minimize time spent by the operator > . Although < cards > is < slightly > better than < dasd > with regard to < additional turnaround time > (a component of < user satisfaction >), < dasd > is < much > better with regard to < the difference in the form the user receives > (a component of < user satisfaction >), < somewhat > better with regard to < clearing time > , and < slightly > better with regard to < additional material cost > .

This is essentially a rehash of the previous response to WHY NOT, except that all attributes are mentioned. The magnitude of the contribution is used to determine the order in which objectives are mentioned.

> EXPLAIN CLEARING TIME

< clearing time > is < the amount of time the plan takes to clear a dataset off the queue > , measured in < minutes > . The < less > < clearing time > , the better. < clearing time > is a component of < effectively manage queue space > .

EXPLAIN combines the information provided by DETAIL and RELEVANCE.

> WHY NOT INSTALL

< dasd > is as good or better than install with regard to every objective which underlies my choice.

This is the case where the chosen alternative is found to dominate another in Step1:

```
dasd:      .1    32.1    1   .5        1
install:   180   180     1   5000      210
           ----------------------------
            +     +      0    +        +
```

Since dasd dominates install, this is simply stated to the user.

4.5 Elements of explanation provided by other facilities

In the previous sections, we have briefly addressed the structure of justifications for choices and mechanisms for generating them. Again, we note that in using the AMVM in the context of ar-

chitectures encompassing other knowledge structures, these justifications must be integrated with the explanation facilities which support these other structures. In explaining actions based on the shallow model of section 2.3, for example, we would want not only to justify the choice of a plan, but additionally to describe the conditions under which the plan is applicable, the steps involved in executing the plan, and perhaps how the plan serves to resolve or circumvent a queue space crisis. But as we discussed in section 2.6, we speculate that these additional elements of explanation could be integrated with UTIL with little or no modification to UTIL itself. For example, to respond to 'WHY NOT *a*', where *a* is currently in the conflict set, an 'explanation supervisor' could simply invoke UTIL's WHY NOT routine. If *a* is not currently in the conflict set, the supervisor might instead invoke a rule explanation routine which describes the conditions under which *a* is applicable and indicates its failed (i.e., unmatched) conditions. In the latter case, it is *a*'s ineligibility which prevents its invocation rather than its desirability.

5. Refining models of choice

The purpose of refinement is threefold. First, refinement may be used to correct erroneous portions of the approximate model captured in the initial acquisition phase. Errors in model the might be due to biased assessments (Tversky & Kahneman (1974, 1981), Hershey et al. (1982)), or to incorrect or carelessly formulated user responses. Second, refinement provides the basis for capturing insights gained from the exercise of initial acquisition, but not reflected in the initial model. Third, and perhaps most important, refinement provides the means to modify the model to reflect changing preferences over time. In short, refinement is a tool for incremental model restructuring.

As previously mentioned, refinement is strongly coupled with explanation. From the prescriptive point of view, UTIL's explanation component serves as a window into how choices are made. In cases where users fail to find UTIL's justifications convincing, they iteratively invoke UTIL's refinement facilities to repair portions of the model until convincing justifications are generated. From the descriptive viewpoint, users may employ UTIL's refinement facilities to capture the underlying basis for what *they* see as the 'best' choice, so that this information may be displayed in explanations for users who did not participate in the formulation of the model. The challenge in building a system for refinement is to provide an environment which makes it convenient for the user to identify and repair precisely those portions of the model that fail to reflect reality.

Our preliminary work in refinement is described as follows. Section 5.1 lists the isolated elements of the AMVM which may need to be changed. Section 5.2 discusses how these changes might be organized by a refinement facility to promote effective model repair. In section 5.3 we present some hypothetical dialogs which encompass the strategies of 5.2, and the mechanisms underlying (UTIL's portion of) their generation are presented in section 5.4. Section 5.5 briefly reflects upon the integration of the described facilities with refinement facilities for other knowledge structures.

5.1 Types of changes

The design of UTIL's refinement facilities is driven by a set of expectations regarding its operation. Specifically, we envision two broadly-defined scenarios in which refinement would be employed. First, we have *offline changes* which occur outside the context of intelligent system operation. For example, a user may add an objective to the model to reflect newly defined standards in the environment (e.g., the EPA issues new standards regarding the disposal of wastes). Users may also change the relationship between objectives in an existing model; for example, as IBM is currently attempting to cut costs across the board, computer installation managers may want to increase the relative importance of objective 'minimize material costs'. Another example, installation management will continually add and delete alternative actions to reflect configuration changes, and these modifications might give rise to new or more detailed objectives that were previously implicit in the model's operation.

Second, we have *online changes* which are motivated by choices generated by the model during actual intelligent system operation. As the user will not examine the explanation for every possible choice after making an offline change, some modifications will undoubtedly be made online.

Both offline and online changes may necessitate one or more modifications to the AMVM:

- changing weights on objectives and subobjectives,
- changing component (attribute) value functions, including modification of returned values for ranges of attribute values and making these ranges more detailed.
- adding objectives or subobjectives,
- adding or modifying context descriptions.

Changes outside, but related to the AMVM include:

- adding alternatives,
- changing the levels of attributes, and
- capturing levels for newly-added attributes.

The challenge of building a refinement facility is to organize these modifications in a way which is most convenient for the user.

5.2 Approach to refinement

In chapter III we briefly described some of the capabilities which we would expect from a refinement facility. In this section, we elaborate on these as an introduction to the sample dialogs of the following sections. In all cases, the goal is to maximize the reliability of model repairs while minimizing the user's effort (itself a problem of tradeoffs among competing objectives). Capabilities of interest include:

1. **Showing the user how proposed changes will affect the ranking of alternatives.** This feedback allows the user to determine if additional refinement is necessary.
2. **Allowing the user to directly correct a suspicious component of an explanation.** The proposed strategy is to allow the user to highlight a portion of an explanation on the screen and to click a FIX button on the mouse (or a PF key or some other input device). UTIL would then invoke the appropriate routine for handling the modification.
3. **Guiding the user through probable errors in the model based on contextual information.** For example, when a newly added alternative is misranked by the model, it is more probable that a new attribute (which (i) distinguishes the new alternative from existing ones, and (ii) is valued approximately equally for existing alternatives) should be added to the model than that existing attribute weights are in error. Another example: When weights on objectives are dramatically changed, it is probable that the user has another decision-making context in mind.
4. **Discouraging the correction of improbable causes for an erroneous choice.** For example, when two alternatives fare equally with regard to a particular attribute, the weight associated with that attribute cannot be held accountable for their incorrect relative ranking. A refinement facility should therefore discourage its correction.
5. **Promoting the reliability of newly-captured parameters.** For example, in capturing a subjective index (e.g., value from 1 to 10), the refinement system should display the extreme values for existing alternatives to give the user a 'feel' for how previous judgements were made.
6. **Isolating subproblems for correction.** The refinement facility should promote the repair of isolated subproblems which may be in error, as organized in the objectives hierarchy.
7. **Inferring or estimating new parameter values where possible.** For example, when a new weight for an objective is captured from the user, one heuristic for inferring new weights for the remaining objectives is to redistribute the remaining weight among them according to their ex-

isting ratios. Such a strategy does not, of course, guarantee correctness, but is a more reasonable alternative than recapturing all weights.

8. **Describing to the user a set of modifications which will result in a particular ranking which he desires.** We would like the proposed system to map users' preferences among alternatives (if these are known) into possible sets of modifications from which the user can select.
9. **Initiating the capture of new information as necessary.** For example, when a new attribute is added to the model, we would expect the refinement facility to initiate the capture of values for this attribute with respect to each existing alternative.

Some of these capabilities (e.g., 8) reflect the descriptive perspective: The user know what the ranking should be and attempts to encode the underlying basis for this ranking for purposes of explanation or prediction. Others (e.g., 2) more reflect the prescriptive view: The user does not know what the ranking should be, and attempts to incrementally provide the information needed to produce a choice which can be satisfactorily justified.

Aspects of these capabilities are illustrated in the following sections in the context of some specific examples.

5.3 Hypothetical dialogs

In this and the following section we present some hypothetical dialogs and hint at implementation methods. A disclaimer: We are not ready to advocate these methods; we will need to implement them and to test them on several examples. They are presented here only to communicate the flavor of the approach we plan to pursue at the outset of the research.

Two dialogs are presented. The first, in which the user controls most of the interaction, illustrates the notion of 'repairing an explanation'. The second, in which UTIL controls most of the interaction, demonstrates the notion of 'guiding the user through refinement'.

As in the section on explanation, we first present the examples in their entirety so as not to obscure the dialog. Following, we reproduce the examples annotated with descriptions of some of the methods we might employ to generate UTIL's end of the interaction.

The syntax of the examples is as follows. User responses follow the prompt symbol (>) and are CAPITALIZED. Text generated by UTIL appears in ordinary case. Annotations are {enclosed in braces}. Assume that UTIL has access to all the information in the sample AMVM of section 1.

5.3.1 User-driven refinement: repairing an explanation

In this section we present an example of what we have described as repairing an explanation. The example also demonstrates the following capabilities:

- Values for the remaining weights are inferred rather than recaptured when one ('minimize material costs') is changed.
- The effects of proposed changes are displayed for the user.
- The isolation of subproblems in the objectives hierarchy is used to infer weights on detailed objectives instead of asking the user to respecify them. Specifically, the component attributes of 'maximize user satisfaction' are automatically recalculated from the newly inferred weight for this attribute. There is no trace of this calculation in the dialog; its absence is the noteworthy point. The calculation is described in the following section, with other mechanisms.

Consider the following (very timely) refinement scenario. IBM is currently making a concerted effort to cut internal costs. In this dialog, an operations manager employs UTIL to increase the

importance of this objective in the AMVM which drives queue space management decisions. The approach is that of displaying the appropriate part of the model using the explanation facility and directly modifying it using the FIX command.

> DETAIL EFFECTIVELY MANAGE QUEUE SPACE

My objective is to effectively manage queue space. In order to achieve that objective I must maximize user satisfaction, minimize material costs, minimize queue clearing time, and minimize time spent by the operator. Assuming that queue space is a little low, maximize user satisfaction is my most important objective, accounting for 50% of the choice. minimize material costs accounts for 20% of the choice, minimize queue clearing time accounts for 20%, and minimize time spent by the operator accounts for the remaining 10%.

{The user highlights the weight for minimize material costs and clicks the FIX key on the mouse}

How much more (e.g., 2 times) or less (e.g., 1/2 times) important is minimize material costs than it used to be?

> 2

O.K. From now on, the choice will be formulated as follows:

minimize material costs is my most important objective, accounting for 40% of the choice. maximize user satisfaction accounts for 37.5% of the choice , minimize queue clearing time accounts for 15%, and minimize time spent by the operator accounts for the remaining 7.5%.

Assuming all known alternatives are eligible for execution, the old model produced the ranking: dasd, expensive-printing, cheap-printing, copy, delete, cards, fiche, install

The new model, taking the increased importance of additional material cost into account, produces the ranking: dasd, cheap-printing, copy, expensive-printing, delete, cards, fiche, install

The difference is that cheap printing and copy are now preferred to expensive printing. Other preferences are as before.

The user might at this point use COMPARE to elicit an explanation as to why, say, cheap printing is preferred to expensive printing in order to verify that material cost is the determining factor.

5.3.2 UTIL-driven refinement

In this section we exemplify UTIL's guidance of the refinement process. As a specific case, we discuss refinement in the context of adding an alternative. In section 3 we presented a similar dialog in which the user was immediately satisfied with the integration. Here, we describe UTIL's response in the case where the user is not satisfied.

When an alternative is added, we will ordinarily need only to capture its underlying attributes; the AMVM should ensure that new alternatives are properly integrated with existing ones. However, if seemingly incorrect results are produced, then the user might attempt to 'modify an explanation' as exemplified in the previous section. This strategy is in accordance with the prescriptive approach to choosing.

Alternatively, if the user is sure that one alternative is better than another and this choice is not in accordance with the AMVM, we want UTIL to guide him in modifying the model in order to

produce the desired result. This strategy is more in accordance with the descriptive approach to choosing. The following dialog illustrates this strategy. Also illustrated are:

- how UTIL might help the user to integrate a new objective into the model;
- how UTIL promotes the reliability of newly captured parameters by displaying relevant existing information. In this example, UTIL displays the attribute values of the best and worst existing alternatives to help the user specify analogous values for the new alternative;
- how UTIL initiates the capture of new information as needed. In this example, UTIL initiates the capture of values for the newly-added objective with respect to existing alternatives;
- how UTIL makes heuristic inferences regarding the probable values of parameters. In this example, UTIL captures the new objective value for only one existing alternative and generalizes it to all alternatives. In addition, weights on objectives are inferred as in the previous section.

In the dialog that follows, let us assume that copy is a new alternative to be added to the existing set. Further assume that 'minimize operator time' is not yet included as an objective. Assume, retaining the same attribute indices (i.e., x2 through x5 exist and we will add x1) as in previous examples for convenience, that our existing value function is:

$$v(x2,x3,x4,x5) = 5/9(.5*v2(x2) + .5*v3(x3)) + 2/9*v4(x4) + 2/9*v5(x5)$$

This value function produces the same ranking on our alternatives (excluding **copy**): dasd, expensive-printing, cheap-printing, delete, cards, fiche, install. We now proceed with the dialog.

> NEW ALTERNATIVE

Please type the name of the alternative.

> COPY

How much additional turnaround time (in minutes) for copy? As a reference point, additional turnaround time for dasd is 32.1 minutes, and is 180 minutes for install.

> 34.2

{Assume that UTIL now captures the values for other attributes in exactly the same fashion, as in section 3.}

Assuming queue space is, say, just a little low, the new ordering on alternatives is:

dasd, copy, expensive-printing, cheap-printing, delete, cards, fiche, install.

Is that correct?

> NO

{If the user did not know if this was correct, he might at this point invoke COMPARE and take control (attempt to repair an explanation). Since his response is NO, UTIL takes over and guides the refinement process}

Sorry. Let's correct the error.

My objective is to effectively manage queue space. In order to achieve that objective I must maximize user satisfaction, minimize material costs, and minimize queue clearing time.

Can you think of some aspect of copy other than user satisfaction, material cost, and queue clearing time which particularly distinguishes it from the remaining alternatives?

> YES

What is it?

> ADDITIONAL OPERATOR TIME

Would we want to

1. maximize or
2. minimize

additional operator time in our attempt to effectively manage queue space ?

> 2

{UTIL would now capture other pieces of information about the objective (e.g., a canned-text definition) which are omitted here for brevity}

Does minimize additional operator time have something to do with

1. maximize user satisfaction,
2. minimize material costs,
3. minimize queue clearing time, or
4. none of the above?

> 4

O.K. Then I will consider minimize additional operator time to be an objective in its own right, to be considered with those mentioned above in order to effectively manage queue space.

Assuming queue space is a little low, how important is additional operator time relative to queue clearing time? For example, user satisfaction is 5 times as important as queue clearing time and material cost is equally as important as queue clearing time.

> 1/2

O.K. As things stand now, I will make choices to effectively manage queue space as follows:

My objective is to effectively manage queue space. In order to achieve that objective I must maximize user satisfaction, minimize material costs, minimize queue clearing time, and minimize additional operator time. Assuming that queue space is a little low, maximize user satisfaction is my most important objective, accounting for 50% of the choice. minimize material costs accounts for 20% of the choice, minimize queue clearing time accounts for 20%, and additional operator time accounts for the remaining 10%.

Does that seem reasonable, for now?

> YES

O.K. Let's talk some more about additional operator time. Is there a quantitative measure (e.g., dollars, minutes) with which we can measure additional operator time ?

> MINUTES

{UTIL would next capture the component value function for the new objective. This is omitted here, for there are several automated methods available}

O.K. How many minutes of additional operator time for the new alternative copy?

> 10

How many minutes of additional operator time for our best alternative, dasd?

> .1

O.K. additional operator time never made much difference before you added copy, so I might as well assume that the other alternatives fare about the same as dasd with regard to additional operator time. Does that sound reasonable?

> YES

The new ranking on alternatives is: dasd, expensive-printing, cheap-printing, copy, delete, cards, fiche, install.

Does that seem reasonable?

> YES

Good.

Had the user still been dissatisfied with the result, UTIL would continue to probe the model with the user (as described in the following section). We would implement such UTIL-guided dialogs such that the user could assume control at any point using the explanation and FIX commands.

5.4 Mechanisms employed in the hypothetical dialogs

In this section we provide a description of some of the mechanisms which might be implemented to produce the dialogs of the previous section.

All responses are generated by instantiating values in response templates. The templates themselves would be selected on the basis of the context of the dialog and of the nature of the explanation appropriate to the situation.

The syntax of the dialog is as follows. As before, user responses follow the prompt symbol (>) and are CAPITALIZED, and text generated by UTIL appears in ordinary case. Variables in response templates are **<enclosed in angle brackets>**. All elements of the dialog are in **bold**, and descriptions of methods used to generate UTIL's responses appear in normal font. Assume that UTIL has access to all the information in the sample AMVM of section 1.

5.4.1 User-driven refinement: repairing an explanation

> DETAIL EFFECTIVELY MANAGE QUEUE SPACE

My objective is to <effectively manage queue space>. In order to achieve that objective I must <maximize user satisfaction>, <minimize material costs>, <minimize queue clearing time>,

and < minimize time spent by the operator >. Assuming that queue space is < a little low >, < maximize user satisfaction > is my most important objective, accounting for < 50 > % of the choice. < minimize material costs > accounts for < 20 > % of the choice, < minimize queue clearing time > accounts for < 20 > %, and < minimize time spent by the operator > accounts for the remaining < 10 > %.

This is accomplished by traversing the top level of the objectives hierarchy and displaying objective names and weights. The user can use DETAIL to explore subobjectives, if desired. Next, the user highlights the weight for 'minimize material costs' and clicks the FIX key on the mouse.

How much more (e.g., 2 times) or less (e.g., 1/2 times) important is < minimize material costs > than it used to be?

> 2

This question is in accordance with the 'ratio method' (see Edwards 1977) of weight determination.

O.K. From now on, the choice will be formulated as follows:

< minimize material costs > is my most important objective, accounting for < 40 > % of the choice. < maximize user satisfaction > accounts for < 37.5 > % of the choice , < minimize queue clearing time > accounts for < 15 > %, and < minimize time spent by the operator > accounts for the remaining < 7.5 > %.

We employ the following method for determining the remaining weights, also inspired by the ratio method. Our objective is to distribute the 'total remaining weight' (in this case 1 - 2*.2 = .6) among the remaining objectives such that the relationships between these objectives are preserved.

The relationships (ratios) of the remaining existing weights are:

- < maximize user satisfaction > is 5 times as important as < minimize time spent by the operator >.
- < minimize queue clearing time > is 2 times as important as < minimize time spent by the operator >.

The redistribution of the remaining weight is:

- < minimize time spent by the operator > thus gets 1/(1 + 2 + 5) = .125 of the remaining weight (.6).
- < minimize queue clearing time > gets 2/(1 + 2 + 5) = .25 of the remaining weight (.6).
- < maximize user satisfaction > gets 5/(1 + 2 + 5) = .625 of the remaining weight (.6).

Thus, the new weights are:

- < minimize time spent by the operator > = .125 * .6 = .075
- < maximize user satisfaction > = .625 * .6 = .375
- < minimize queue clearing time > = .25 * .6 = .15

Note that the ratios have been preserved:

- < maximize user satisfaction > = .375 = 5 * .075 = 5 * < minimize time spent by the operator >.
- < minimize queue clearing time > = .15 = 2 * .075 = 2 * < minimize time spent by the operator >.

The new weights for top-level objectives are propagated down the objectives hierarchy to calculate the new weights for detailed attributes:

- < additional turnaround time > = .5 (objective weight) * .375 (new user satisfaction weight) = .1875
- < difference in form > = .5 (objective weight) * .375 (new user satisfaction weight) = .1875

This yields the new value function:
v(x1,x2,x3,x4,x5)
= .075*v1(x1) + .375(.5*v2(x2) + .5*v3(x3)) + .4*v4(x4) + .15*v5(x5)
= .075*v1(x1) + .1875*v2(x2) + .1875*v3(x3) + .4*v4(x4) + .15*v5(x5)

Note that the relationship between new weights for detailed attributes remains as before (i.e., equal). Also note that their new values have been excluded from the display of new weights, since refinement occurred at the top-level in the objectives hierarchy.

Assuming all known alternatives are eligible for execution, the old model produced the ranking: < dasd, expensive-printing, cheap-printing, copy, delete, cards, fiche, install >

The new model, taking the < increased > importance of < additional material cost > into account, produces the ranking: < dasd, cheap-printing, copy, expensive-printing, delete, cards, fiche, install >

The difference is that < cheap printing > and < copy > are now preferred to < expensive printing >. Other preferences are as before.

This can be implemented by computing the rankings with the old and newly modified value functions, and determining their differences.

5.4.2 UTIL-driven refinement

The first issue in UTIL-driven refinement concerns the ordering of model components to explore with the user. One approach is to employ *scripts* which express heuristic strategies for refinement. A script is simply an ordered set of refinement actions (e.g., attempt to capture a new objective, attempt to revise weights) appropriate to a particular refinement situation (e.g., a new alternative was just added). UTIL would follow these scripts in order to guide refinement. Note that no particular script is guaranteed to minimize the user's effort; rather, scripts organize potential refinement actions in terms of the most likely causes for error as justified by heuristic arguments.

For example, in the case of a newly-added alternative which is incorrectly ranked, it is probable that an objective is missing from the hierarchy, one which has the following characteristics: (i) the objective serves to distinguish the new alternative's (dis)value with respect to existing alternatives, and (ii) existing alternatives fare approximately equally with regard to the missing objective, since it's consideration was never before required to produce correct rankings. 'Minimize operator time', for instance, is an important objective in all operator's minds, but it is not explicitly considered unless one or more alternatives differ from the rest in terms of it. Continuing with the script, if a new objective is added but the alternatives are still incorrectly ranked, we might next focus on the utility function(s) associated with the new objective, since these have never before been tested and refined, and so on. Relationships between existing objective weights are probably least likely to be incorrect, since these have presumably been used in prior choice situations with existing alternatives. We would employ a very different script in the situation where an erroneous choice is identified in the context of an existing model. In this case, erroneous tradeoffs (weights) are among the more likely candidates for correction. We will, of course, need to experiment with various scripts for various situations.

An alternative to scripts (which we have not yet thought much about) involves *inferring* the most likely causes of error rather than encoding them. Heuristics for directing refinement under various conditions might be specified in rules such as: IF a new parameter has recently been specified AND the choice is incorrect THEN reassess that parameter.

Finally, we might formulate the choice of which portion of the model to verify/repair first as a multiattribute value problem. The model would encompass objectives such as 'minimize user effort' and 'minimize age of model portion' (i.e., repair newer portions first).

We now proceed to describe mechanisms which might support the other capabilities mentioned in section 5.3.2, assuming the use of scripts.

> NEW ALTERNATIVE

Please type the name of the alternative.

> COPY

UTIL prompts the user for the values of the attributes in the AMVM. This is trivially accomplished by referencing the names of attributes in the hierarchy. UTIL would also capture other pieces of information such as a description of the component plan step which most influences its value, but these are omitted here for brevity.

How much < additional turnaround time > (in < minutes >) for < copy > ? As a reference point, < additional turnaround time > for < dasd > is < 32.1 > < minutes > , and is < 180 > < minutes > for < install > .

> 34.2

Reference points are provided by simple lookup for the best and worst alternatives according to the current value function. This is done so that the user can see how liberal or conservative the estimates are for existing quantitative attributes and to give the user a feel for relative values in the case of subjective indices.

Having collected attribute values, UTIL shows the user how the new alternative has been integrated into the existing set; that is, UTIL displays the new ranking of alternatives, assuming all are simultaneously eligible for execution. A context (amount of space left) is selected for display. Alternatively, we might display multiple rankings, each corresponding to a known context.)

Assuming queue space is, say, < just a little low > , the new ordering on alternatives is:

< dasd, copy, expensive-printing, cheap-printing, delete, cards, fiche, install > .

This is trivially accomplished by presenting the alternatives as ordered by value function evaluation.

Is that correct?

> NO

Sorry. Let's correct the error.

UTIL now begins to follow the script for 'add an alternative'. The first step is to identify a missing objective and to capture its related characteristics. Note that there is nothing in the system which *guarantees* that new attributes and objectives satisfy the independence assumptions upon which the

additive form is based. At this point we might therefore ask the user some standard questions to verify that the independence assumptions hold (Keeney & Raiffa 1976), but these are omitted here for brevity.

My objective is to < effectively manage queue space >. In order to achieve that objective I must < maximize user satisfaction >, < minimize material costs >, and < minimize queue clearing time >.

Can you think of some aspect of < copy > other than < user satisfaction >, < material cost >, and < queue clearing time > which particularly distinguishes it from the remaining alternatives?

> YES

What is it?

> ADDITIONAL OPERATOR TIME

Would we want to

1. **maximize or**
2. **minimize**

< additional operator time > in our attempt to < effectively manage queue space > ?

> 2

UTIL would now capture other pieces of information about the objective (such as a canned description) which are omitted here for brevity.

UTIL now establishes the position of the new objective in the objectives hierarchy.

Does < minimize additional operator time > have something to do with

1. **< maximize user satisfaction >,**
2. **< minimize material costs >,**
3. **< minimize queue clearing time >, or**
4. **none of the above?**

> 4

O.K. Then I will consider < minimize additional operator time > to be an objective in its own right, to be considered with those mentioned above in order to < effectively manage queue space >.

UTIL now knows that < additional operator time > is a top-level objective, so it attempts to capture its weight. Had the new objective been a component objective of one of the three mentioned, UTIL would probe further to find its appropriate place in the objectives hierarchy.

Assuming queue space is < a little low >, how important is < additional operator time > relative to < queue clearing time >? For example, < user satisfaction > is < 5 times > as important as < queue clearing time > and < material cost > is < equally > as important as < queue clearing time >.

> 1/2

O.K. As things stand now, I will make choices to < effectively manage queue space > as follows:

My objective is to < effectively manage queue space >. In order to achieve that objective I must < maximize user satisfaction >, < minimize material costs >, < minimize queue clearing time >, and < minimize additional operator time >. Assuming that queue space is < a little low >, < maximize user satisfaction > is my most important objective, accounting for < 50 > % of the choice. < minimize material costs > accounts for < 20 > % of the choice, < minimize queue clearing time > accounts for < 20 > %, and and < additional operator time > accounts for the remaining < 10 > %.

Does that seem reasonable, for now?

> YES

Using the method of section 5.4.1, UTIL computes the new weights for existing objectives, maintaining their existing ratios. UTIL now captures the subhierarchy associated with the new objective.

O.K. Let's talk some more about < additional operator time >. Is there a quantitative measure (e.g., < dollars >, < minutes >) with which we can measure < additional operator time > ?

> MINUTES

UTIL now knows that < minimize additional operator time > has no lower level objectives. Had the user answered NO, UTIL would prompt for additional objectives until the user either identified a quantitative measure for each or could not think of any more lower-level objectives, at which point a subjective index would be suggested.

UTIL would next capture the component value function for the new objective. This is omitted here, for there are several automated methods available.

UTIL must now capture the attribute values for existing alternatives which pertain to the new objective.

O.K. How many < minutes > of < additional operator time > for the new alternative < copy > ?

> 10

How many < minutes > of < additional operator time > for our best alternative, < dasd > ?

> .1

O.K. < additional operator time > never made much difference before you added < copy >, so I might as well assume that the other alternatives fare about the same as < dasd > with regard to < additional operator time >. Does that sound reasonable?

> YES

Since the new objective did not serve to distinguish the existing set of alternatives, its value is assumed to be the same across alternatives. Thus, UTIL only needs to capture one value. In this example, it turns out this assumption is valid for all alternatives except **install**. If it is later discovered (during system operation) that this is unacceptable, it can be corrected then. Had the user answered NO to UTIL's last question, UTIL would prompt for the exceptions and their respective values. Thus, we adopt the approach of minimizing the user's effort up front, assuming that he may continue to tune the model either immediately or over time. The alternative strategy would

have been to explicitly capture values for the new objective across all alternatives, a potentially wasteful and effort-intensive prospect for the user.

The new ranking on alternatives is: dasd, expensive-printing, cheap-printing, copy, delete, cards, fiche, install

Does that seem reasonable?

> YES

Good.

5.5 Elements of refinement provided by other facilities

In previous sections, we have briefly addressed the refinement of choice models and related mechanisms in the context of a 'top-level' model. Again, we note that in using the AMVM in the context of some more encompassing architecture, these methods must be integrated with the refinement facilities which support other elements of the architecture.

In the shallow model of section 2.3, for example, we would want not only to refine the knowledge underlying the choice of a plan, but additionally to modify the plans themselves and the sets of conditions under which they are applicable. But as we discussed in section 2.6, it seems that these additional elements of refinement could be integrated with UTIL without significant modification to UTIL itself. A higher level 'refinement supervisor' would be necessary to coordinate the various elements of refinement corresponding to different structures.

Chapter VI: Research Plan

We will proceed as follows in continuing the work of chapter V toward the achievement of the goals of chapter III.

We will begin with explanation, developing a minimal set of facilities that seem appealing and encompass the discourse elements identified in chapter V. Next, we will begin implementing the refinement facilities. As explanation and refinement are strongly coupled, this will undoubtedly uncover additional requirements for explanation. We will continue to iterate between explanation and refinement, using queue space management as the vehicle domain, until a stable, appealing set of facilities are produced.

The core implementation will be done in a Symbolics environment in LISP. Specialized software may be employed for constructing interfaces as the need arises.

There are essentially three dimensions to demonstrating the value of our results:

- *practicality:* We will need to demonstrate the usefulness of our facilities for practical problems. Toward this end, we will employ (and improve upon, if necessary) our formulation of the queue space management domain, and extensively test UTIL using this domain.
- *domain-independence:* We need to show that our results are sufficiently general to apply to domains beyond queue space management. To accomplish this, we will test UTIL in some toy domains. Likely candidates include choosing among competing entrees at dinner and choosing an activity for a Saturday evening. The potential usefulness of such systems will be considered irrelevant; domains will be chosen according to ease of knowledge engineering and ease of understandability by the general population who might read the thesis.
- *architecture-independence:* While clearly a worthwhile endeavor, we *do not* propose to implement the sort of architectures sketched in chapter V. On the other hand, we do not wish to completely ignore the issue of integration with other knowledge structures. Thus, we will speculate on this issue, sketching the potential strategies for integration. This will force us to keep modularity in mind in developing UTIL. Integration will be framed as a logical extension of the research to be completed after graduation.

We will be informally evaluating UTIL throughout its development. While a formal evaluation would be highly desirable, this is another task which we propose to perform after graduation because of the time involved. Our plans for (informal) evaluation include reviewing work-in-progress with the committee members, with colleagues, and with the computer operators at IBM Research who would presumably use UTIL.

Chapter VII: Research Contributions

As the proposed thesis addresses the synthesis of ideas from two historically distinct fields (broadly identified here as *artificial intelligence* and *decision analysis*), we separately discuss its potential contributions to each in sections 1 and 2 respectively.

1. Contributions to artificial intelligence

We can identify two broad areas of contribution for the proposed work. Our primary contributions will be in the areas of automated explanation and knowledge acquisition, as this is among the first efforts to provide facilities for explaining and refining decision-theoretic choices. In particular, UTIL will be the first domain-independent, integrated facility that we know of for explaining and refining multiattribute value models.

Successful completion of the work will also represent a contribution to knowledge-based systems. As decision-analytic models are useful for making complex choices in the context of such systems, providing domain-independent facilities for explaining and refining these models should encourage their employment in knowledge-based architectures. Briefly addressing the integration of facilities for explaining and refining decision-analytic models with analogous facilities for other knowledge structures should provide added encouragement. In addition, the discussion of the AMVM-based architecture in section 2.3 of chapter V suggests a general framework for fusing procedural, rule-based, and choice knowledge in intelligent systems where such representations are useful. While we will not have time to implement this architecture, a slightly more detailed sketch would provide a useful foundation for further work. In particular, our casting of the queue space management domain in this architecture represents a step toward developing a general framework for building more effective intelligent process control systems.

2. Contributions to decision analysis

We can identify several related contributions to decision analysis resulting from the successful completion of UTIL.

First, the automated explanation of choices is novel to decision analysis. As we mentioned, UTIL is among the first such efforts. This, and efforts such as (Langlotz et al. 1986) may play an important role in influencing the behavior of decision makers. By providing decision makers with convincing justifications, we may be able to 'open the black box', thereby increasing the acceptance of decision-theoretic results.

Second, UTIL represents one of the first attempts to provide for incremental problem restructuring in the context of the decision-theoretic paradigm. Such facilities are crucial for interactively capturing changing preferences over time, and this is a virtually unexplored area in the literature.

The provision of facilities for incremental problem restructuring also has implications for other areas of decision analysis research. Specifically, UTIL may help to reduce the performance-related demands on initial acquisition methods, for if we are armed with the means for incrementally tuning decision-analytic models, we need not build flawless models on the first iteration of model construction. In particular, UTIL represents a novel approach to handling the biases of initial acquisition methods. Thus, the successful construction of UTIL might impact research in related fields of decision analysis.

The use of decision analysis for intelligent process control is also novel. While decision-analytic models have been used in a large variety of settings, this is the first application of which we are

aware for making realtime decision-theoretic choices in the environments of complex physical system control rooms.

In summary, UTIL addresses issues in both artificial intelligence and decision analysis. We believe that its successful completion would represent a contribution to both fields.

References

1. Aikens, J., 'Prototypes and production rules: a knowledge representation for computer consultations', PhD thesis, Stanford University, 1980.
2. Astrom, K., Anton, J., Arzen, K., 'Expert control', *Automatica*, Vol. 22, No. 3, 1986.
3. Barrow, H., 'VERIFY: A program for proving correctness of digital hardware designs', *Artificial Intelligence 24*, 1984.
4. Bell, D., 'Multiattribute utility functions: decompositions using interpolations', *Management Science 25*, 1979.
5. Ben-Bassat, M., Carlson, R., Puri, V., Davenport, M., Schriver, J., Latif, M., Smith, R., Portigal, L., Lipnick, E., Weil, M., 'Pattern-based interactive diagnosis of multiple disorders: the MEDAS system', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol PAMI-2, No 2, 1980.
6. Berliner, H., Ackley, D., 'The QBKG system: generating explanations from a non-discrete knowledge representation', *Proceedings of AAAI*, 1982.
7. Bobrow, D., ed., 'Qualitative reasoning about physical systems', Elsevier, 1985.
8. Boyd, D., Clark, C., North, D., 'Analysis of a government initiative to accelerate commercial acceptance of solar photovoltaic systems', technical report, Decision Focus Inc., Palo Alto, California, 1982.
9. Brown, R., Kahr, A., Peterson, C., 'Decision analysis for the manager', Holt, 1974.
10. Chandrasekaran, B., Mittal, S., 'Deep verses compiled knowledge approaches to diagnostic problem solving', *International Journal of Man-Machine Studies*, 1983.
11. Chester, D., Lamb, D., Dhurjati, P., 'Rule based alarm analysis in chemical process plants', *Proc. Micro-Delcon*, 1984.
12. Chou, C., Cruise, A., Finkel, A., Klein, D., Loeb, D., Masullo, M., Milliken, K., Waite, N., 'Improved RETE Pattern Matching', *Proceedings of the IBM Expert Systems ITL*, 1986.
13. Clancey, W., 'Methodology for building an intelligent tutoring system', STAN-CS-81-894, Stanford University, 1981.
14. Clancey, W., 'Details of the revised therapy algorithm', in Buchanan, B. & Shortliffe, E., eds, *Rule-based expert systems*, Addison-Wesley, 1984.
15. Cohen, M., Axelrod, R., 'Coping with complexity: the adaptive value of changing utility', *American Economic Review 74*, 1984.
16. Coles, L., Robb, A., Sinclair, P., Smith., M., Sobek, R., 'Decision analysis for an experimental robot with unreliable sensors', *Proceedings of IJCAI*, 1975.
17. Cromarty, A., 'What are current expert system tools missing?', *Proceedings of COMPCOM*, 1985.
18. Cruise, A., Ennis, R., Finkel, A., Loeb, D., Masullo, M., Milliken, K., Van Woerkom, H., Waite, N., 'YES/MVS - An expert system for the automation of large system operation', *Proceedings of European Guide*, 1986.
19. Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Klein, D., Loeb, D., Masullo, M., Milliken, K., Van Woerkom, H., Waite, N., 'YES/MVS and the automation of operations for large computer complexes', *IBM Systems Journal 25*, 1986.
20. Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Loeb, D., Klein, D., Masullo, M., Milliken, K., Van Woerkom, H., Waite, N., 'Isolating functionality in realtime expert systems', Proceedings of the IBM Expert Systems ITL, 1986.
21. Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Klein, D., Loeb, D., Masullo, M., Milliken, K., Van Woerkom, H., Waite, N., 'YES/L1: integrating rule-based, procedural, and realtime programming for industrial applications', *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, 1987.
22. Cyert, R., DeGroot, M., 'Adaptive utility', in Day, R., Groves, T., eds., *Adaptive Economic Models*, Academic Press, 1975.

23. Davis, R., 'Teiresias: applications of meta-level knowledge', PhD Thesis, Stanford University, 1976. Reprinted in Davis, R., Lenat, D., *Knowledge-based systems in artificial intelligence*', McGraw Hill, 1982.
24. Davis, R., Buchanan, B., 'Meta-level knowledge: overview and applications', *Proceedings of IJCAI*, 1977.
25. Davis, R., 'Meta-rules: reasoning about control', *Artificial Intelligence* 15, 1980.
26. Davis, R., 'Diagnostic reasoning based on structure and behavior', *Artificial Intelligence 24*, 1984.
27. Dawes, R., 'The robust beauty of improper linear models in decision making', *American Psychologist 34:7*, 1979.
28. Debreu, G., 'Topological methods in cardinal utility theory', in *Mathematical methods in the social sciences*, Arrow, K., Karlin, S., Suppes, P., eds., Stanford University Press, 1960.
29. DeJong, K., 'Intelligent control: integrating AI and control theory', *IEEE Conference on Trends and Applications*, 1983.
30. de Kleer, J., Brown, J., 'A qualitative physics based on confluences', *Artificial Intelligence 24*, 1984.
31. de Neufville, R., Keeney,R., 'Use of decision analysis in airport development in Mexico City', in *Analysis of Public Systems*, Drake, W., Keeney, R., Morse, P., eds., MIT Press, 1972.
32. Dyer, J., Sarin, R., 'Measurable multiattribute value functions', *Operations Research 27*, 1979.
33. Edwards, W., 'How to use multiattribute utility theory for social decision making', *IEEE Transactions on Systems, Man, and Cybernetics 7*, 1977.
34. Edwards, W., 'Reflections on and criticisms of a highly political multiattribute utility analysis', in *Mathematical Frontiers of Behavioral and Policy Sciences*, Cobb, L., Thrall, R., eds., Westview Press, 1980.
35. Ennis, R., Griesmer, J., Hong, S., Karnaugh, M., Kastner, J., Klein, D., Milliken, K., Schor, M., Van Woerkom, H., 'YES/MVS: a continuous realtime expert system', *Proceedings of AAAI*, 1984.
36. Ennis, R., Griesmer, J., Hong, S., Karnaugh, M., Kastner, J., Klein, D., Milliken, K., Schor, M., Van Woerkom, H., 'Automation of MVS operations, an expert systems approach', *Proceedings of Computer Measurement Group Conference XV*, 1984. Also in *Computer Systems Science and Engineering*, Vol I, No I.
37. Ennis, R., Griesmer, J., Hong, S., Karnaugh, M., Klein, D., Milliken, K., Schor, M., Van Woerkom, H., 'A computer operator's expert system', *Proceedings of the Seventh International Conference on Computer Communications*, 1984.
38. Ennis, R., Griesmer, J., Hong, S., Karnaugh, M., Kastner, J., Klein, D., Milliken, K., Schor, M., Van Woerkom, H., 'A continuous realtime expert system for computer operations', *IBM Journal of Research and Development*, Vol. 30, No. 1, 1986. Reprinted in *Data Processing 28*, 1986.
39. Farquhar, P., Fishburn, P., 'Equivalence and continuity in multivalent preference structures', *Operations Research 29*, 1981.
40. Farquhar,P., 'Utility assessment methods', *Management Science 30*, 1984.
41. Farquhar, P., 'Applications of utility theory in artificial intelligence research', TR 86-2, Graduate School of Industrial Administration, Carnegie-Mellon University, 1986.
42. Feldman, J., Sproull, R., 'Decision theory and artificial intelligence II: the hungry monkey', *Cognitive Science 1*, 1975.
43. Ferrante, R., 'The characteristic error approach to conflict resolution', *Proceedings of IJCAI*, 1985.
44. Fink, P., 'Control and integration of diverse knowledge in a diagnostic expert system', *Proceedings of IJCAI*, 1985.
45. Fishburn, P., 'Methods of estimating additive utilities', *Management Science 13*, 1967.
46. Fishburn, P., 'Utility theory for decision making', Wiley, 1970.
47. Forbus, K., Stevens, A., 'Using qualitative simulation to generate explanations', *Proceedings of the Third Annual Conference of the Cognitive Science Society*, 1981.
48. Forbus, K., 'Qualitative process theory', *Artificial Intelligence 24*, 1984.

49. Forgy, C., 'OPS5 user's manual', CMU-CS-81-135, Carnegie-Mellon University, 1981.
50. Forgy, C., 'OPS83 User's Manual', Dept. of Computer Science, Carnegie-Mellon University, 1984.
51. Friedman,L., 'Controlling production rule firing: the FCL language', *Proceedings of IJCAI*, 1985.
52. Georgeff,M., 'Procedural control in production systems', *Artificial Intelligence* 18, 1982.
53. Good, I., 'Good thinking: the foundations of probability and its applications', University of Minnesota Press, 1983.
54. Gorry, G., Silverman, H., Pauker, S., 'Capturing clinical expertise: a computer program that considers clinical responses to digitalis', *American Journal of Medicine 64*, 1978.
55. Green, P., Srinivasan, V., 'Conjoint analysis in consumer research: issues and outlook', *Journal of Consumer Research*, Vol 5, 1978.
56. Hart, P., 'Directions for AI in the eighties', *SIGART Newsletter* No. 79, January 1982.
57. Hershey, J., Kunreuther, H., Schoemaker, P., 'Sources of bias in assessment procedures for utility functions', *Management Science 28*, 1982.
58. Holloway, C., 'Decision making under uncertainty', Prentice-Hall, 1979.
59. Howard, R., Matheson, J., North, D., 'The decision to seed hurricanes', *Science 176*, 1972.
60. Huber, G., 'Methods for quantifying subjective probabilities and multiattribute utilities', *Decision Science 5*, 1974.
61. Jacobs, W., Keifer, M., 'Robot decisions based on maximizing utility', *Proceedings of IJCAI*, 1973.
62. Jungermann, H., 'Structural modelling of decision problems', Institute for Psychology, Technical University of Berlin, 1980.
63. Kass, R., Finin, T., 'Modelling the user in natural language systems', *Computational Linguistics*, special issue on user modelling, 1987.
64. Kastner, J., 'Strategies for expert consultation in therapy planning', PhD Thesis, Rutgers University, 1983.
65. Kaufman, G, Thomas, H., eds., 'Modern decision analysis', Penguin Books, 1977.
66. Keefer, D., Kirkwood, C., 'A multiobjective decision analysis: budget planning for project engineering', *Journal of the Operational Research Society 29*, 1978.
67. Keelin, T., 'A parametric representation of additive value functions', *Management Science 27*, 1981.
68. Keen, P., Scott Morton, M., 'Decision support systems: an organizational perspective', Addision-Wesley, 1978.
69. Keeney, R., 'Examining corporate policy using multiattribute utility analysis', *Sloan Management Review 17*, 1975.
70. Keeney, R., Raiffa, H., 'Decisions with multiple objectives: preferences and value tradeoffs', Wiley, 1976.
71. Keeney, R., Sicherman, A., 'Assessing and analyzing preferences concerning multiple objectives: an interactive computer program', *Behavioral Science 21*, 1976.
72. Keeney, R., 'Siting energy facilities', Academic Press, 1980.
73. Keeney, R., 'Analysis of preference dependencies among objectives', *Operations Research 29*, 1981.
74. Keeney, R., 'Decision analysis: an overview', *Operations Research 30*, 1982.
75. Keeney, R., 'Value-driven expert systems', *Proc. Multi-Attribute Decision Making via OR-Based Expert Systems*, 1986.
76. Kirkwood, C., Sarin, R., 'Preference conditions for multiattribute value functions', *Operations Research 28*, 1980.
77. Klein, D., Milliken, K., 'YES/MVS: managing large installations with expert systems', *Proceedings of SHARE 63*, 1984.
78. Klein, D., 'An expert systems approach to realtime, active management of a target resource', MBA/MSE thesis, MS-CIS-85-40, University of Pennsylvania, 1985.
79. Klein, D., 'Qualitative reasoning about physical systems: theory, practice, and problems', Area Exam in CIS, University of Pennsylvania, 1986.

80. Klein, D., Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Loeb, D., Masullo, M., Milliken, K., Van Woerkom, H., Waite, N., 'YES/L1: Integrating expert systems technology with traditional programming languages', *Proceedings of the IBM Expert Systems ITL*, 1986.
81. Klein, D., Finin, T., (a) 'On the requirements of active expert systems', TR MS-CIS-87-03, University of Pennsylvania. Also in to appear in *Proceedings of AVIGNON-87: Seventh International Conference on Expert Systems and their Applications*, May, 1987.
82. Klein, D., Finin, T., (b) 'What's in a deep model? A characterization of knowledge depth in intelligent safety systems', *Proceedings of IJCAI*, to appear August 1987.
83. Klein, G., Moskowitz, S., Ravindran, A., 'Simplified assessment of single- and multi-attribute utility functions via mathematical programming', Report 82-7, Dept. of MIS, University of Arizona, 1982.
84. Kosey, E., Wise, B., 'Self-explanatory financial planning models', *Proceedings of AAAI*, 1984.
85. Koton, P., 'Empirical and model-based reasoning in expert systems', *Proceedings of IJCAI*, 1985.
86. Krantz, D., 'Conjoint measurement: the Luce-Tukey axiomatization and some extensions', *Journal of Mathematical Psychology 1*, 1964.
87. Krantz, D., Luce, R., Suppes, P., Tversky, A., 'Foundations of measurement', Academic Press, 1971.
88. Krischer, J., 'An annotated bibliography of decision analytic applications to health care', *Operations Research 28*, 1980.
89. Langlotz, C., Fagan, L., Tu, S., Williams, J., Sikic, B., 'ONYX: An architecture for planning in uncertain environments', *Proceedings of IJCAI*, 1985.
90. Langlotz, C., Shortliffe, E., Fagan, L., 'Using decision theory to justify heuristics', *Proceedings of AAAI*, 1986.
91. Langlotz, C., Shortliffe, E., Fagan, L., 'A methodology for generating computer-based explanations of decision-theoretic advice', working paper KSL-86-57, Depts. of medicine and computer science, Stanford University, 1986.
92. Lanthrop, J., Watson, S., 'Decision analysis for evaluation of risk in nuclear waste management', *Journal of the Operational Research Society 33*, 1982.
93. LaValle, I., 'On cash equivalents and information evaluation in decisions under uncertainty', *Journal of the American Statistical Association 63*, 1968.
94. LaValle, I., 'Fundamentals of decision analysis', Holt, 1978.
95. Lehner, P., Probus, M., Donnell, M., 'Building decision aids: exploiting the synergy between decision analysis and artificial intelligence', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-15, No 4, 1985.
96. LISP/VM User's Guide, Order #SH20-6477-0, IBM Corporation, 1984.
97. Luce, R., Raiffa, H., 'Games and decisions', Wiley, 1957.
98. Luce, R., Tukey, J., 'Simultaneous conjoint measurement: a new type of fundamental measurement', *Journal of Mathematical Psychology 1*, 1964.
99. Madni, A., Samet, M., Purcell, D., 'Adaptive models in information management', in Andriole, S., ed., *Applications in Artificial Intelligence*, Petrocelli, 1985.
100. Magee, J., 'How to use decision trees in capital investment', *Harvard Business Review 42*, 1964.
101. McDermott, J., Forgy, C., 'Production system conflict resolution strategies', in *Pattern-directed inference systems*, Hayes-Roth, F., Waterman, D., eds., Academic Press, 1978.
102. Merkhofer, M., 'The value of information given decision flexibility', *Management Science*, 1977.
103. Meyer, R., 'On the relationship among the utility of assets, the utility of consumption, and investment strategy in an uncertain, but time invariant world', *OR 69: Proceedings of the Fifth International Conference on Operational Research*, 1970.
104. Michalski, R., Carbonell, J., Mitchell, T., eds., 'Machine learning: an artificial intelligence approach', Tioga, 1983.
105. Michalski, R., Carbonell, J., Mitchell, T., eds., 'Machine learning: an artificial intelligence approach', Volume II, Tioga, 1986.

106. Miller, P., 'Critiquing anesthetic management: the ATTENDING computer system', *Anesthesiology 58*, 1983.
107. Milliken, K., 'Using expert system technology to automatically assist MVS operation', *Proceedings of GUIDE 59*, 1984.
108. Milliken, K., Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Loeb, D., Klein, D., Masullo, M., Van Woerkom, H., Waite, N., 'Flexibility in the YES/MVS knowledge base', *Proceedings of the IBM Expert Systems ITL*, 1985.
109. Moore, P., Thomas, H., 'The anatomy of decisions', Penguin Books, 1976.
110. Moore, R., Hawkinson, L., Knickerbocker, C., Churchman, L., 'A real-time expert system for process control', *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE/AAAI, 1984.
111. Nair, K., Sicherman, A., 'Environmental assessment methodology: solar power plant applications, volume 4: decision analysis computer program', report ER-1070, Electric Power Research Institute, Palo Alto, CA, 1979.
112. Novick, M., Issacs, G., Hamer, R., Chen, J., Chuang, D., Woodworth, G., Molenaar, I., Lewis, C., Libby D., 'Manual for the computer-assisted data analysis monitor', University of Iowa, Iowa City, 1980.
113. Nelson, W., 'REACTOR: an expert system for diagnosis and treatment of nuclear reactor accidents', *Proceedings of AAAI*, 1982.
114. Nilsson, N., *Principles of Artificial Intelligence*, Tioga, 1980.
115. North, D., Offensend, F., Smart, C., 'Planning wildlife protection for the Santa Monica mountains', *Fire Journal 69*, 1975.
116. O'Leary, D., 'Multiple criteria decision making in accounting expert systems', *Sixth International Workshop on Expert Systems and their Applications*, 1986.
117. Pratt, J., Raiffa, H., Schlaifer, R., 'The foundations of decision under uncertainty: an elementary exposition', *Journal of the American Statistical Association 59*, 1964.
118. Pratt, J., Raiffa, H., Schlaifer, R., 'Introduction to statistical decision theory', McGraw-Hill, 1965.
119. Ramsey, F., 'Truth and probability', a 1926 lecture published in *The Foundations of Mathematics and Other Logical Essays*, Humanities Press, 1950.
120. Raiffa, H., 'Decision analysis', Addison-Wesley, 1968.
121. Ray, W., 'Advanced process control', McGraw-Hill, 1981.
122. Reggia, J., Perricone, B., 'Answer justification in medical decision support systems based on Bayesian classification', *Comp. Biol. Medicine 15*, 1985.
123. Rennels, G., Shortliffe, E., Miller, P., 'A model of choice and explanation in medical management', *Medical Decision Making*, 1987.
124. Rich, E., 'User modeling via stereotypes', *Cognitive Science 3*, 1979.
125. Rich, E., 'Users are individuals: individualizing user models', *Int. J. Man-Machine Studies 18*, 1983.
126. Sarin, R., 'Ranking of multiattribute alternatives with an application to coal power plant siting', in *Multiple Criteria Decision Making - Theory and Application*, Fandel, G., Gal, T., eds., Springer-Verlag, 1980.
127. Sauers, R., Walsh, R., 'On the requirements of future expert systems', *Proceedings of IJCAI*, 1983.
128. Savage, L., 'The foundations of statistics', Wiley, 1954.
129. Scarl, E., Jamieson, J. Delaune, C., 'A fault detection and isolation method applied to liquid oxygen loading for the space shuttle', *Proceedings of IJCAI*, 1985.
130. Schlaifer, R., 'Analysis of decisions under uncertainty', McGraw-Hill, 1969.
131. Schlaifer, R., 'Computer programs for elementary decision analysis', Harvard Business School, 1971.
132. Schor, M., Daly, T., Lee, H.S., Tibbitts, B., 'YES/OPS Extensions to OPS5: Language and Environment', IBM RC 11900 (#53025), March 1986.
133. Seo, F., Sakawa, M, Takanashi, H., Nakagami, K., Horiyama, H., 'An interactive computer program for multiattribute utility analysis', GE18-1980-0, IBM Tokyo Scientific Center, 1978.

134. Shortliffe, E., 'Computer-based medical consultations: MYCIN', Elsevier/North Holland, 1976.
135. Slagle, J., Hamburger, H., 'An expert system for a resource allocation problem', *Communications of the ACM 28:9*, 1985.
136. Spronk, J., Zionts, S., 'A special issue on multicriteria decision-making', *Management Science 30*, 1984.
137. Stephanopoulous, G., 'Chemical process control', Prentice Hall, 1984.
138. Stevens, A., 'STEAMER: Advanced computer aided instruction in propulsion engineering', BBN Technical Report #4702, 1981.
139. Stillwell, W., Barron, F., Edwards, W., 'Evaluating credit applications: a validation of multiattribute utility techniques against a real world criterion', SSRI Research Report 80-1, UCLA, 1980.
140. Swartout, W., 'Producing explanations and justifications of expert consultation programs', PhD thesis, MIT, 1981.
141. Tamura, H., Nakamura, Y., 'Decompositions of multiattribute utility functions based on a new concept of convex dependence', *Conference of IEEE Systems, Man, and Cybernetics Society*, 1978.
142. Teach, R., Shortliffe, E., 'An Analysis of Physicians' Attitudes', *Computers in Biomedical Research 14*, 1981. Reprinted in Buchanan, B., Shortliffe, E., eds., *Rule-Based Expert Systems*, Addison-Wesley, 1985.
143. Tribus, M., 'Rational descriptions, decisions, and designs', Pergamon Press, 1969.
144. Tversky, A., Kahneman, D., 'Judgment under uncertainty: heuristics and biases', *Science 185*, 1974.
145. Tversky, A., Kahneman, D., 'The framing of decisions and the psychology of choice', *Science 211*, 1981.
146. Ulvila, J., Snider, W., 'Negotiation of international oil tanker standards: an application of multiattribute value theory', *Operations Research 28*, 1980.
147. von Neumann, J., Morgenstern, O., *Theory of Games and Economic Behavior*, Princeton University Press, 1947.
148. von Nitzsch, R., Weber, M., 'Utility function assessment on a micro-computer: a reliable, interactive procedure', working paper 86/02, Institut fuer Wirtschaftswissenschaften, RWTH Aachen, Germany, 1986.
149. von Winterfeldt, D., 'Structuring decision problems for decision analysis', *Acta. Psychol. 45*, 1980.
150. von Winterfeldt, D., 'Setting standards for offshore oil discharges: a regulatory decision analysis', *Operations Research 30*, 1982.
151. von Winterfeldt, D., Edwards, W., 'Decision analysis and behavioral research', Cambridge University Press, 1986.
152. Waterman, D., 'A guide to expert systems', Addison-Wesley, 1986.
153. Weber, M., 'A method of multiattribute decision making with incomplete information', *Management Science*, Vol 31, No 11, 1985.
154. Weber, M., 'Decision making with incomplete information', *European Journal of Operational Research 28*, 1987.
155. Weiner, J., 'BLAH, a system which explains its reasoning', *Artificial Intelligence 15*, 1980.
156. Weld, D., 'Explaining complex engineering devices', BBN Technical Report #5489, 1984.
157. Wellman, M., 'Reasoning about preference models', MIT-LCS-TR-340, MIT, 1985.
158. White III, C., Sykes, E., 'A user preference guided approach to conflict resolution in rule-based expert systems', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-16, No 2, 1986.
159. Winkler, R., 'Introduction to Bayesian inference and decision', Holt, 1972.
160. Winograd, T., 'Understanding natural language', Academic Press, 1972.
161. Wright, P., Bourne, D., Colyer, J., Schatz, G., Isasi, J., 'A flexible manufacturing cell for swaging', *Mechanical Engineering*, 1982.
162. Zeleny, M., 'Multiple criteria decision making', McGraw-Hill, 1982.

106. Miller, P., 'Critiquing anesthetic management: the ATTENDING computer system', *Anesthesiology 58*, 1983.
107. Milliken, K., 'Using expert system technology to automatically assist MVS operation', *Proceedings of GUIDE 59*, 1984.
108. Milliken, K., Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Loeb, D., Klein, D., Masullo, M., Van Woerkom, H., Waite, N., 'Flexibility in the YES/MVS knowledge base', *Proceedings of the IBM Expert Systems ITL*, 1985.
109. Moore, P., Thomas, H., 'The anatomy of decisions', Penguin Books, 1976.
110. Moore, R., Hawkinson, L., Knickerbocker, C., Churchman, L., 'A real-time expert system for process control', *Proceedings of the First Conference on Artificial Intelligence Applications*, IEEE/AAAI, 1984.
111. Nair, K., Sicherman, A., 'Environmental assessment methodology: solar power plant applications, volume 4: decision analysis computer program', report ER-1070, Electric Power Research Institute, Palo Alto, CA, 1979.
112. Novick, M., Issacs, G., Hamer, R., Chen, J., Chuang, D., Woodworth, G., Molenaar, I., Lewis, C., Libby D., 'Manual for the computer-assisted data analysis monitor', University of Iowa, Iowa City, 1980.
113. Nelson, W., 'REACTOR: an expert system for diagnosis and treatment of nuclear reactor accidents', *Proceedings of AAAI*, 1982.
114. Nilsson, N., *Principles of Artificial Intelligence*, Tioga, 1980.
115. North, D., Offensend, F., Smart, C., 'Planning wildlife protection for the Santa Monica mountains', *Fire Journal 69*, 1975.
116. O'Leary, D., 'Multiple criteria decision making in accounting expert systems', *Sixth International Workshop on Expert Systems and their Applications*, 1986.
117. Pratt, J., Raiffa, H., Schlaifer, R., 'The foundations of decision under uncertainty: an elementary exposition', *Journal of the American Statistical Association 59*, 1964.
118. Pratt, J., Raiffa, H., Schlaifer, R., 'Introduction to statistical decision theory', McGraw-Hill, 1965.
119. Ramsey, F., 'Truth and probability', a 1926 lecture published in *The Foundations of Mathematics and Other Logical Essays*, Humanities Press, 1950.
120. Raiffa, H., 'Decision analysis', Addison-Wesley, 1968.
121. Ray, W., 'Advanced process control', McGraw-Hill, 1981.
122. Reggia, J., Perricone, B., 'Answer justification in medical decision support systems based on Bayesian classification', *Comp. Biol. Medicine 15*, 1985.
123. Rennels, G., Shortliffe, E., Miller, P., 'A model of choice and explanation in medical management', *Medical Decision Making*, 1987.
124. Rich, E., 'User modeling via stereotypes', *Cognitive Science 3*, 1979.
125. Rich, E., 'Users are individuals: individualizing user models', *Int. J. Man-Machine Studies 18*, 1983.
126. Sarin, R., 'Ranking of multiattribute alternatives with an application to coal power plant siting', in *Multiple Criteria Decision Making - Theory and Application*, Fandel, G., Gal, T., eds., Springer-Verlag, 1980.
127. Sauers, R., Walsh, R., 'On the requirements of future expert systems', *Proceedings of IJCAI*, 1983.
128. Savage, L., 'The foundations of statistics', Wiley, 1954.
129. Scarl, E., Jamieson, J. Delaune, C., 'A fault detection and isolation method applied to liquid oxygen loading for the space shuttle', *Proceedings of IJCAI*, 1985.
130. Schlaifer, R., 'Analysis of decisions under uncertainty', McGraw-Hill, 1969.
131. Schlaifer, R., 'Computer programs for elementary decision analysis', Harvard Business School, 1971.
132. Schor, M., Daly, T., Lee, H.S., Tibbitts, B., 'YES/OPS Extensions to OPS5: Language and Environment', IBM RC 11900 (#53025), March 1986.
133. Seo, F., Sakawa, M, Takanashi, H., Nakagami, K., Horiyama, H., 'An interactive computer program for multiattribute utility analysis', GE18-1980-0, IBM Tokyo Scientific Center, 1978.

134. Shortliffe, E., 'Computer-based medical consultations: MYCIN', Elsevier/North Holland, 1976.
135. Slagle, J., Hamburger, H., 'An expert system for a resource allocation problem', *Communications of the ACM 28:9*, 1985.
136. Spronk, J., Zionts, S., 'A special issue on multicriteria decision-making', *Management Science 30*, 1984.
137. Stephanopoulous, G., 'Chemical process control', Prentice Hall, 1984.
138. Stevens, A., 'STEAMER: Advanced computer aided instruction in propulsion engineering', BBN Technical Report #4702, 1981.
139. Stillwell, W., Barron, F., Edwards, W., 'Evaluating credit applications: a validation of multiattribute utility techniques against a real world criterion', SSRI Research Report 80-1, UCLA, 1980.
140. Swartout, W., 'Producing explanations and justifications of expert consultation programs', PhD thesis, MIT, 1981.
141. Tamura, H., Nakamura, Y., 'Decompositions of multiattribute utility functions based on a new concept of convex dependence', *Conference of IEEE Systems, Man, and Cybernetics Society*, 1978.
142. Teach, R., Shortliffe, E., 'An Analysis of Physicians' Attitudes', *Computers in Biomedical Research 14*, 1981. Reprinted in Buchanan, B., Shortliffe, E., eds., *Rule-Based Expert Systems*, Addison-Wesley, 1985.
143. Tribus, M., 'Rational descriptions, decisions, and designs', Pergamon Press, 1969.
144. Tversky, A., Kahneman, D., 'Judgment under uncertainty: heuristics and biases', *Science 185*, 1974.
145. Tversky, A., Kahneman, D., 'The framing of decisions and the psychology of choice', *Science 211*, 1981.
146. Ulvila, J., Snider, W., 'Negotiation of international oil tanker standards: an application of multiattribute value theory', *Operations Research 28*, 1980.
147. von Neumann, J., Morgenstern, O., *Theory of Games and Economic Behavior*, Princeton University Press, 1947.
148. von Nitzsch, R., Weber, M., 'Utility function assessment on a micro-computer: a reliable, interactive procedure', working paper 86/02, Institut fuer Wirtschaftswissenschaften, RWTH Aachen, Germany, 1986.
149. von Winterfeldt, D., 'Structuring decision problems for decision analysis', *Acta. Psychol. 45*, 1980.
150. von Winterfeldt, D., 'Setting standards for offshore oil discharges: a regulatory decision analysis', *Operations Research 30*, 1982.
151. von Winterfeldt, D., Edwards, W., 'Decision analysis and behavioral research', Cambridge University Press, 1986.
152. Waterman, D., 'A guide to expert systems', Addison-Wesley, 1986.
153. Weber, M., 'A method of multiattribute decision making with incomplete information', *Management Science*, Vol 31, No 11, 1985.
154. Weber, M., 'Decision making with incomplete information', *European Journal of Operational Research 28*, 1987.
155. Weiner, J., 'BLAH, a system which explains its reasoning', *Artificial Intelligence 15*, 1980.
156. Weld, D., 'Explaining complex engineering devices', BBN Technical Report #5489, 1984.
157. Wellman, M., 'Reasoning about preference models', MIT-LCS-TR-340, MIT, 1985.
158. White III, C., Sykes, E., 'A user preference guided approach to conflict resolution in rule-based expert systems', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-16, No 2, 1986.
159. Winkler, R., 'Introduction to Bayesian inference and decision', Holt, 1972.
160. Winograd, T., 'Understanding natural language', Academic Press, 1972.
161. Wright, P., Bourne, D., Colyer, J., Schatz, G., Isasi, J., 'A flexible manufacturing cell for swaging', *Mechanical Engineering*, 1982.
162. Zeleny, M., 'Multiple criteria decision making', McGraw-Hill, 1982.