Int. J. Man–Machine Studies (1987) **26,** 3–28

Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems

JOHN H. BOOSE AND JEFFREY M. BRADSHAW

Knowledge Systems Laboratory, Boeing Advanced Technology Center 77–64, Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124, U.S.A.

Acquiring knowledge from a human expert is a major problem when building a knowledge-based system. Aquinas, an expanded version of the Expertise Transfer System (ETS), is a knowledge-acquisition workbench that combines ideas from psychology and knowledge-based systems research to support knowledge-acquisition tasks. These tasks include eliciting distinctions, decomposing problems, combining uncertain information, incremental testing, integration of data types, automatic expansion and refinement of the knowledge base, use of multiple sources of knowledge and providing process guidance. Aquinas interviews experts and helps them analyse, test, and refine the knowledge base. Expertise from multiple experts or other knowledge sources can be represented and used separately or combined. Results from user consultations are derived from information propagated through hierarchies. Aquinas delivers knowledge by creating knowledge bases for several different expert-system shells. Help is given to the expert by a dialog manager that embodies knowledge-acquisition heuristics.

Aquinas contains many techniques and tools for knowledge acquisition; the techniques combine to make it a powerful testbed for rapidly prototyping portions of many kinds of complex knowledge-based systems.

Obtaining and modeling expertise

EXPERTISE TRANSFER SYSTEM

The Expertise Transfer System (ETS) has been in use in Boeing for more than 3 years. Hundreds of prototypical knowledge-based systems have been generated by ETS. The system interviews experts to uncover key aspects of their problem-solving knowledge. It helps build very rapid prototypes (typically in less than 2 h), assists the expert in analysing the adequacy of the knowledge for solving the problem, and creates knowledge bases for several expert system shells (S.1, M.1, OPS5, KEE, and so on) from its own internal representation (Boose, 1984, 1985, 1986).

The tools in ETS are now part of Aquinas, a much larger system. Aquinas was developed to overcome ETS's limitations in knowledge representation and reasoning (Fig. 1). Due to these limitations, ETS was usually abandoned sometime during the knowledge-acquisition process. Typically project approaches were explored or feasibility was assessed for several days or a week, and then development continued in some other expert system shell. While the use of the tool in this way saved substantial time (typically 1 or 2 calendar months from a 12–24-month project), it was desirable to explore new approaches for making the system more powerful.

0020-7373/87/010003 + 26$03.00/0

Features of AQUINAS

Improved process efficiency:
- Rapid feasibility analysis;
- Multiple alternative testing with little resource expenditure;
- Expert enthusiasm;
- Easier to learn expert-system and knowledge engineering concepts;
- Group knowledge elicitation and decision making.

Faster knowledge-base generation:
- Very rapid prototyping;
- Vocabulary identification;
- Solution elicitation;
- Trait elicitation through triads and other methods;
- Hierarchies,
  - Problem decomposition;
  - Reasoning at varied levels of abstraction.

Improved knowledge-base quality:
- Embedded testing and feedback during the knowledge elicitation process;
- Multiple knowledge representations;
- Multiple methods for handling uncertainty based on needed precision, convenience;
- Tools for comparing knowledge from different experts to show similarities and differences;
- Consultation systems giving consensus and dissenting opinions from multiple sources of knowledge;
- Analytic tools.

Better knowledge-base maintenance and comprehensibility:
- Case-based and knowledge source-based elicitation, structure, analysis;
- Knowledge at higher levels of abstraction;
- Single central source generation of expert-system shell knowledge bases;
- Knowledge libraries.

Extensions to personal construct theory methods:
- Manipulation of rating grids in hierarchies;
- Multiple variable scale types;
- Many analytic tools in a single framework;
- Interactive testing and debugging of rating grid knowledge.

FIG. 1. Aquinas is a knowledge-acquisition workbench that provides a variety of capabilities.

AQUINAS TASKS AND TOOL SETS

Aquinas is a collection of integrated tool sets. They share a common user interface (the dialog manager) and underlying knowledge representation and data base (Fig. 2). Each set of tools addresses a general knowledge-acquisition task and embodies sets of strategies that support the task. Many of these strategies will be illustrated later.

TASK: ELICIT DISTINCTIONS

Gaines (in press) has characterized knowledge acquisition as: "the modeling of events enabling adequate prediction and action". In this view, a *distinction* is the primitive concept underlying the representation of knowledge and the formal theory of modeling. Systems that acquire problem-solving knowledge seek to establish qualitative and quantitative distinctions that lead to effective prediction and action, while weeding out distinctions that are redundant or inconsequential.

Eliciting distinctions with Aquinas

Personal construct psychology. George Kelly's personal construct theory (Kelly, 1955) provides a rich framework for modeling the qualitative and quantitative

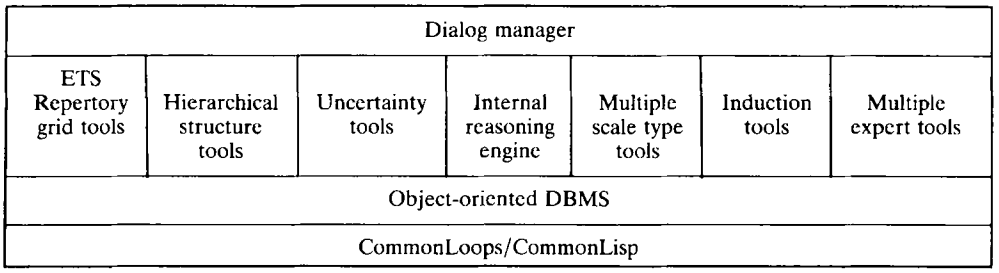

FIG. 2. The Aquinas workbench is a collection of integrated tool sets that support various knowledge-acquisition tasks.

distinctions inherent in an expert's problem solving knowledge. Expertise Transfer System (ETS) is a set of tools used by the expert to elicit, analyse, and refine knowledge as rating grids. In a rating grid, problem solutions—*elements*—are elicited and placed across the grid as column labels, and traits of these solutions—*constructs*—are listed alongside the rows of the grid (Fig. 3, taken from the Programming Language Advisor). Traits are first elicited by presenting groups of solutions and asking the expert to discriminate among them. Following this, the expert gives each solution a rating showng where it falls on the trait scale.

Many of the strategies used in building a rating grid are extensions of ideas in the work of Kelly and in the PLANET system (Gaines & Shaw, 1981; Shaw & Gaines, in press *a, b*). These strategies include triadic elicitation, corner filling, and multiple

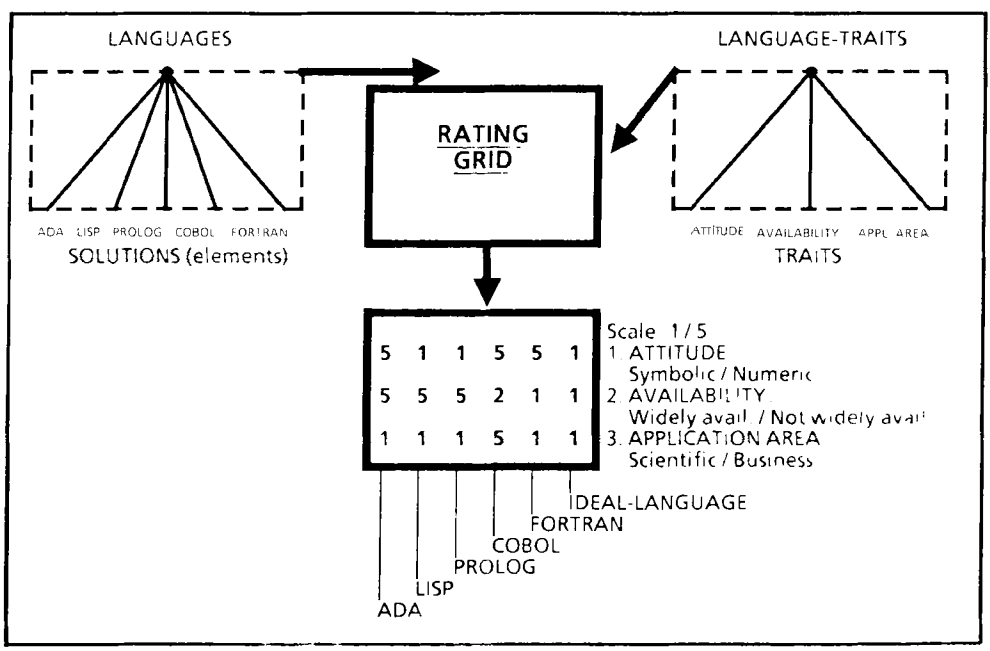

FIG. 3. Rating values in different hierarchies combine to form *rating grids*. The children of a node in a *solution hierarchy* supply the solutions along the top of the grid; the children of a node in a *trait hierarchy* supply the traits down the side of a grid.

analysis and display tools. Aquinas can analyse a rating grid in many ways to help the expert refine useful distinctions and eliminate those that are inconsequential or redundant. Distinctions captured in grids can be converted to other representations such as production rules, fuzzy sets, or networks of frames.

TASK: DECOMPOSE PROBLEMS

Experts building large knowledge bases face the task of decomposing their problem in ways that enhance efficiency and clarity. In our previous work using ETS, the difficulty of representing complex problems in a single rating grid became clear. First, a single rating grid can represent only "flat" relations between single solutions and traits. No deep knowledge, causal knowledge, or relationship chains can be shown. A second limitation was that only solutions or traits at the same level of abstraction could be used comfortably in a single grid. Finally, large single grids were often difficult to manipulate and comprehend.

Problem decomposition strategies in Aquinas

Hierarchies. Hierarchical tools in Aquinas help the expert build, edit, and analyse knowledge in hierarchies and lattices. These hierarchies allow the expert to break up complex problems into pieces of convenient size and similar levels of abstraction. Hierarchies in Aquinas are organized around *solutions, traits,* knowledge sources (i.e. *experts*), and *cases.*

Nodes in the four hierarchies combine to form rating grids. In the most simple case, the children of a node in a solution hierarchy supply the solutions along the top of a grid; the children of a node in a trait hierarchy supply the traits down the side of a grid. Rating values within the grid provide information about the solutions with respect to each trait (Fig. 3).

In eliciting knowledge for complex problems it is sometimes difficult for the expert to identify conclusion sets whose members are at similar, useful levels of granularity. For instance, in an engine diagnostic system, the expert may include the repair solutions "engine", "battery", "ignition coil" and "electrical system". "Engine" and "electrical system" are at more general levels of structural and functional abstraction than "battery" and "ignition coil". Mixing more general and more specific solutions in the same rating grid causes problems during trait elicitation, since traits useful in differentiating "engine" from "electrical system" problems are not necessarily those useful in discriminating "ignition coil" from "battery" problems.

Solution hierarchies. Solutions are grouped in specialization hierarchies within Aquinas. This structure aids experts in organizing large numbers of solutions that may exist at different levels of abstraction. For example, a solution class named "vehicle" is a superclass (parent or prototype) to "car" and "truck" subclasses. The "car" class can serve in turn as a parent to a class of specific car models or to a particular instance of a car.

Trait hierarchies. Characteristics of a particular level in the solution hierarchy can be structured in trait hierarchies. For instance, in a knowledge base for a Transportation Advisor, the solutions exist in hierarchies of vehicles. Each level in the solution hierarchy has a trait hierarchy that contains information needed to select solutions at that level. A trait hierarchy attached to the "vehicle" abstraction level of a solution hierarchy, for instance, may contain information about general

use type, relative speed, cost, and so forth for the types of vehicles in the hierarchy. The "car" subclass is attached to a car trait hierarchy that contains information useful in selecting a particular car.

Two other hierarchies are formed in Aquinas (Fig. 4).

Expert hierarchies. Expert hierarchies represent multiple knowledge sources as structured groups. Each node in the expert hierarchy may represent an individual, an aspect of an individual, a group, or an independent knowledge source. Information from multiple experts may be independently elicited and analysed, then weighted and combined to derive joint solutions to problems. Analyses can be performed that show similarities and differences among experts. Experts each have their own solution and trait hierarchies, which may or may not overlap those of others. Each expert's unique problem-solving strategies and information are preserved.

Case hierarchies. Case hierarchies define subsets of the knowledge base appropriate to solving a particular class of problems. For example, in a knowledge base of information about vehicles, a user may want to include different knowledge for selecting a vehicle for going over land than for going over water. A land case and a water case may be created, each drawing on a subset of the expert pool knowledgeable in those areas. Additional levels may be created for short or long land trips, cost considerations, and so on. A hierarchy of cases allows the knowledge base to be developed, modified, and maintained based on specific classes of situations. Eventually the lower leaves in case hierarchies become specific consultation instances when the knowledge is tested and used to solve a specific problem.

From hierarchies to rating grids. A rating grid is built by combining values associated with nodes in each of the four basic hierarchies. Relationships between nodes do not have to be strictly hierarchical; lattices may be formed when more than

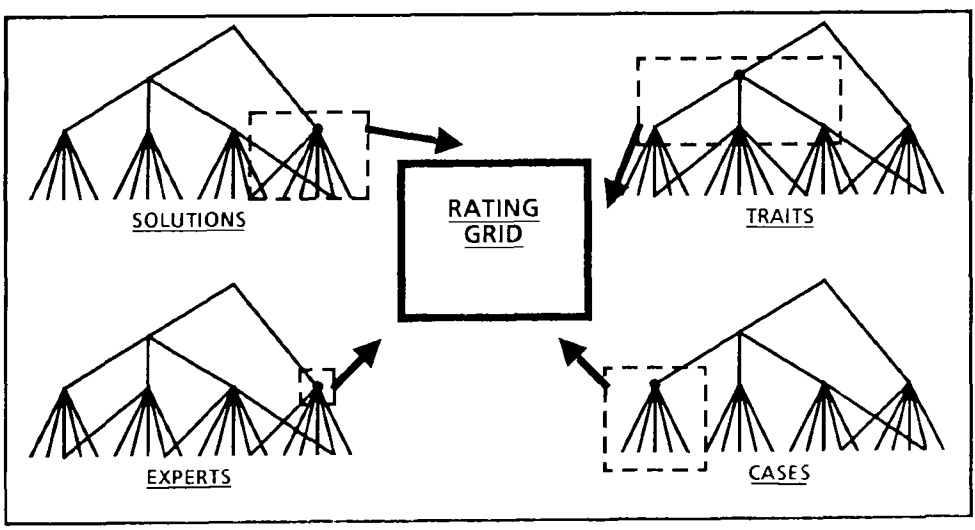

FIG. 4. Values from *expert* and *case hierarchies* as well as solution and trait hierarchies are combined in many ways to form rating grids. Relationships between nodes do not have to be strictly hierarchical; lattices may be formed when more than one parent points to the same child.

one parent points to the same child. The expert defines the current rating grid by selecting appropriate nodes in the hierarchies.

Figure 5 shows selected map nodes (case: K-ACQUISITION; expert: WEC; solution: WEC.ELEMENT; trait: WEC.ELEMENT.TRAIT) that define the rating grid of Fig. 3. Each different collection of nodes (at least one from each hierarchy) describes a rating grid. A rating grid could be a single column or row, or even a single cell. Inversely, each cell in a rating grid is uniquely described by its location in the four hierarchies.

In a sense, each rating grid is four-dimensional. Any two of these dimensions are shown at once as rows and columns in a given grid. Usually solutions and traits are shown, but sometimes it is useful to show other combinations. For instance, a grid could display the ratings of several experts across the top with particular solutions down the side. The associated trait and case nodes would be shown to the side of the grid. Often the ratings displayed summarize or generalize information from different nodes in the hierarchies; this issue is discussed later.

Techniques for defining and exploring hierarchies. Strategies for helping the expert build and refine hierarchies in Aquinas include laddering, cluster analysis, and trait value examination. Some of these strategies will be demonstrated in the section describing the Programming Language Advisor.

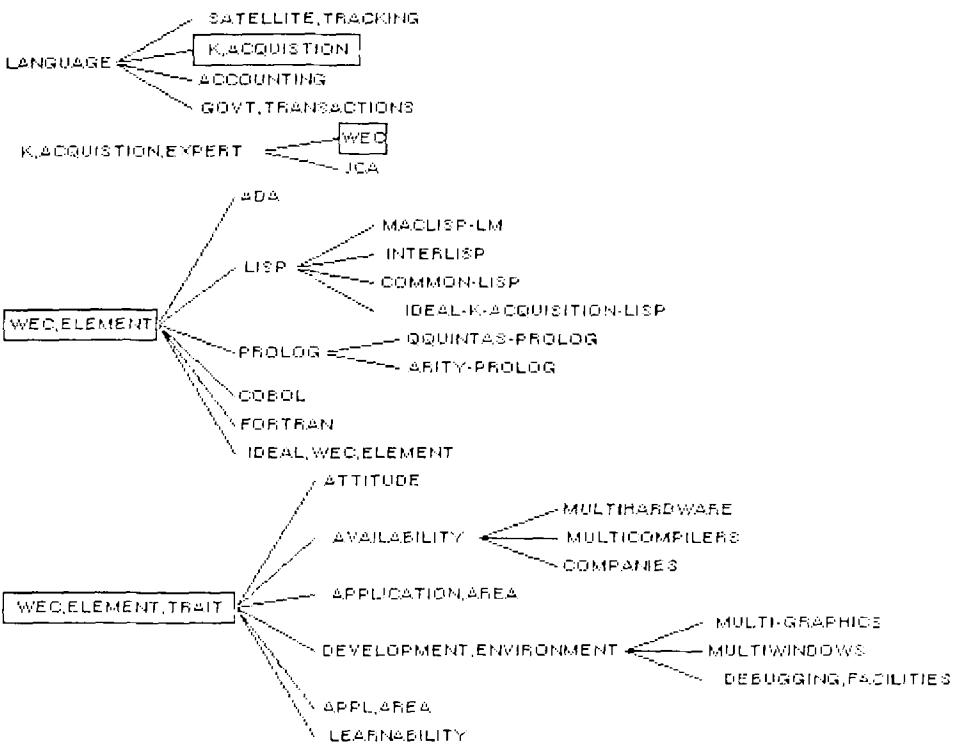

FIG. 5. Each cell in a rating grid is described by a unique set of four hierarchy nodes. Aquinas users specify rating grids by selecting sets of nodes (either the nodes themselves or their children).

TASK: SPECIFY METHODS FOR COMBINING UNCERTAIN INFORMATION

A major limitation of most current knowledge engineering tools is that they do not allow experts to specify how specific pieces of information should be combined (Gruber & Cohen, in press). Most tools either tend to use fixed, global numeric functions to compute values or restrict the expert to purely symbolic representations of uncertainty. Ideally, flexibility and comprehensibility could be achieved by allowing experts to specify how information should be combined locally either by selecting from a set of commonly accepted combining functions (e.g. as done by Reboh & Risch, 1986; Reboh, Risch, Hart & Duda, 1986) or by defining their own method.

Combining uncertain information in Aquinas

In Aquinas, uncertain knowledge, preferences, and constraints may be elicited, represented, and locally applied using combinations of several different methods. These methods may be classified into three main types: absolute, relativistic, and probabilistic.

Absolute reasoning. Absolute (categorical) reasoning involves judgments made with no significant reservations. It "typically depends on relatively few facts, its appropriateness is easy to judge, and its result is unambiguous" (Szolovitz & Pauker, 1978). For example, in selecting a programming language, users may be able to say with certainty that they would be interested only in languages that run on an Apple Macintosh or that they will not consider a language that costs more than $400, regardless of other desirable characteristics. Experts can also build these types of absolute constraints into the knowledge associated with an Aquinas rating grid.

Relativistic reasoning. Unfortunately, not all judgments can be absolute. Many involve significant trade-offs, where information and preferences from several sources must be weighed. Even if criteria for the ideal decision can be agreed on, sometimes it can be only approximated by the available alternatives. In these cases, problem-solving information must be propagated in a relativistic fashion. Aquinas incorporates a variety of models and approaches to relativistic reasoning, including MYCIN-like certainty factor calculus (Adams, 1985), fuzzy logic (Gaines & Shaw, 1985), and the Analytic Hierarchy Process (AHP, Saaty, 1980).

Probabilistic and user-defined reasoning. In the current version of Aquinas, some limited propagation of probabilistic information is made possible by allowing discrete distributions on rating values. Future versions of Aquinas will have more complete models for the elicitation (Alpert & Raiffa, 1982; Spetzler & Stal von Holstein, 1984; Wallsten & Budescu, 1983) and analysis of probabilistic information including Bayesian (Howard and Matheson, 1984; Cheeseman, 1985; Henrion, 1986; Pearl 1986; Spiegelhalter, 1986), Dempster–Shafer (Shafer, 1976; Gordon & Shortliffe, 1985), and other approaches (Shastri & Feldman, 1985). Users may also define their own methods for combining and propagating information.

The availability of different inference methods within a single workbench allows users and experts flexibility in adapting Aquinas to the problem at hand. Methods are currently selected based on the cost of elicitation, the precision of the knowledge needed, convenience, and the expert's preference. Future research will suggest heuristics for helping experts select appropriate methods and designs for particular

types of questions (e.g. Shafer & Tversky, 1985). These heuristics will be incorporated into the Aquinas dialog manager.

TASK: TEST THE KNOWLEDGE

McDermott (1986) has emphasized the inseparability of acquired knowledge from the role it plays in problem solving. Within a given knowledge-acquisition tool, the problem method must be available to the expert as the knowledge base is being constructed so that incremental testing and refinement can take place.

Testing knowledge in Aquinas

A mixed-initiative reasoning engine within Aquinas supports consultations. The model of problem solving currently used in Aquinas is that of multiple knowledge sources (experts) that work together in a common problem-solving context (case) by selecting the best alternatives for each of a sequential set of decisions (solutions). Alternatives at each step are selected by combining relevant information about preferences (relativistic reasoning), constraints (absolute reasoning) and evidence (probabilistic reasoning).

For many structured selection problems, a more specialized version of this model seems adequate. After analysing several expert systems for classification, Clancey (1986) suggested that many problems are solved by abstracting data, heuristically mapping higher-level problem descriptions onto solution models, and then refining these models until specific solutions are found (Fig. 6). This is also similar to the establish-refine cycle used in CSRL (Bylander & Mittal, 1986; Chandrasekaran, 1986; Bylander & Chandrasekaran, in press). In the version of Aquinas described in this paper, data abstraction is carried out within hierarchies of traits, and solutions are refined as information is propagated through solution hierarchies.

While the current version of Aquinas works best on those problems whose solutions can be comfortably enumerated (such as those amenable to the method of heuristic classification), we are interested in generalizing Aquinas to incorporate synthetic (constructive) problem-solving methods such as those in SALT (Marcus, in press).

TASK: INTEGRATE DIVERSE DATA TYPES

Problem solving in knowledge-based systems often involves combining symbolic and numeric information. Qualitative and quantitative aspects are complimentary rather

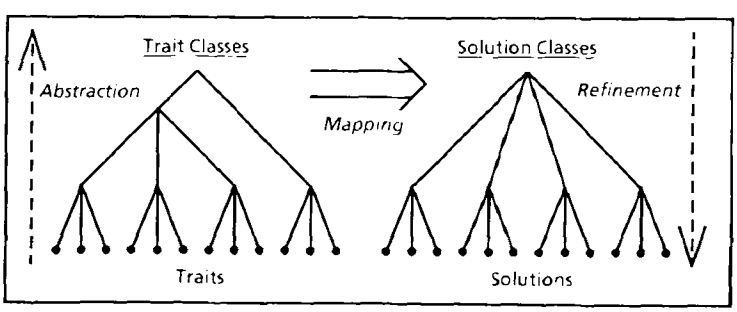

FIG. 6. Clancey (1986) studied structured selection systems and built an abstraction and refinement model. Inference in Aquinas typically occurs in a bottom–up fashion through the trait hierarchies and in top–down fashion through the solution hierarchies.

than opposing considerations, so knowledge-acquisition tools need to represent such information flexibily and conveniently. In our work with ETS, we found that it was inconvenient to represent certain types of problem-solving information solely using Kelly's constructs. Unordered variables, such as a set of computer types, had to be represented as a series of bipolar traits (VAX/NOT-VAX, IBM/NOT-IBM, and so on) when it would have been easier to combine them into a single nominal trait (a COMPUTER trait whose values are VAX, IBM, and so on).

Experts also apply different levels of precision at different points in the knowledge-acquisition process. For example, in some instances it might be sufficient to know that an object is hot or cold. At a later point, it may be important to know the exact temperature of the object. Levels of precision must also be appropriately flexible. ETS only dealt with ordinal ratings on a scale from 1 to 5, not probabilities or exact numeric values.

Integrating data types within Aquinas

In Aquinas, various trait (attribute) scale types can be elicited, analysed, and used by the reasoning engine. Traits are currently described according to the level of measurement of their rating scales, which is determined by the expert. The level of measurement depends on the presence or absence of four characteristics: *distinctiveness, ordering in magnitude, equal intervals,* and *absolute zero* (Coombs, Dawes & Tversky, 1970). These four characteristics describe the four major levels of measurement, or types of traits: nominal (unordered symbols), ordinal, interval, and ratio (Fig. 7). The additional information about trait types gives increased power to analytical tools within Aquinas and allows experts to represent information at the level of precision they desire.

Ratings may be generated through several methods:

(1) Direct. An expert directly assigns a rating value for a trait and an element. If an exact value is unknown, Aquinas helps the expert derive an estimate (Beyth-Marom & Dekel, 1985). If fine judgments are needed, Aquinas can derive a set of ratio scaled ratings from a series of *pairwise comparisons* (Saaty, 1980). Aquinas also contains tools for encoding of probability distributions on specific values. The value with the highest probability is displayed in the grid, but all appropriate values are used in reasoning and may be edited with graphic distribution aids.

(2) Derived. Incomplete grids can be automatically filled through propagation of

RATING SCALE	DESCRIPTION	EXAMPLES
Nominal	Unordered set	LANGUAGE: {ADA COBOL LISP}
Ordinal	Ordered set	COLD/HOT: {1 2 3 4 5} SIZE: {SMALL MEDIUM LARGE}
Interval	Ordered set with measurable intervals	SMALL-INTEGERS: {1 2 3 4 5 6 7} F-TEMP: {32 . . 112}
Ratio	Ordered set with measurable intervals and an absolute origin	HEIGHT: {0.0'1.0' . .}

FIG. 7. Aquinas expands the knowledge representation capability of rating grids from personal construct theory by allowing the use of several types of rating scale values. Scale types are selected for convenience, precision, or efficiency of value entry.

rating values from another grid through the hierarchies (e.g. from lower-to higher-level grids, different experts, or different cases).

Precision and cost. Increased precision and specificity in knowledge acquisition allow increased problem-solving power but usually at some cost (Michalski & Winston, 1985). This cost is reflected in both the amount of work needed to elicit the additional information and increased complexity and greater number of steps in the reasoning process. Aquinas tries to minimize this cost by eliciting more precise information only when it is needed to solve critical portions of the problem. If, for example, Aquinas finds that it cannot sufficiently discriminate between solutions from simple rating values between 1 and 5, it may suggest that the user perform a series of pairwise comparisons to increase the sensitivity of judgments.

TASK: AUTOMATIC EXPANSION AND REFINEMENT OF THE KNOWLEDGE BASE

Knowledge-acquisition tools can increase their leverage by suggesting appropriate expansions and refinements of the knowledge based on partial information already provided by the expert. Michalski (1986) has discussed the advantages of incorporating learning strategies within conventional knowledge-acquisition tools.

Expanding and refining knowledge with Aquinas

Several types of tools make inductive generalizations about existing knowledge. Generalizations can be examined by the expert and used to refine the knowledge, and are used by the reasoning engine. Sometimes, Aquinas may suggest that traits be deleted after analysing the knowledge through a process that is similar to the simplification of decision tables (Hurley, 1983, Michalski, 1978) and decision trees (Quinlan, 1983).

Learning strategies in Aquinas include simple learning from examples (e.g. selective induction on lower level grids to derive values for higher level grids), deduction (e.g. inheritance of values from parents), analogy (e.g. derivation of values based on functional similarity of traits), and observation (e.g. constructive induction based on cluster analysis). The dialog manager (described below) also contains various learning mechanisms.

TASK: USE MULTIPLE SOURCES OF KNOWLEDGE

Future knowledge-acquisition systems can neither assume a single source of expertise nor a closed world. In ETS, we began experimenting with strategies for manually combining ETS knowledge from several domain experts (Boose, in press). Others in our laboratory have been involved in developing methods for cooperative problem-solving (Benda, Jagannathan & Dodhiawala, 1986).

Using multiple sources of knowledge in Aquinas

Knowledge from multiple experts (or other knowledge sources) can be analysed to find similarities and differences in knowledge, and the degree of subsumption of one expert's knowledge over another (Gaines & Shaw, 1981). Information from analyses can be used to guide negotiation among experts. The reasoning engine uses knowledge from user-specified and weighted sources and gives consensus and dissenting opinions.

TASK: PROVIDE KNOWLEDGE-ACQUISITION PROCESS GUIDANCE

As knowledge-acquisition tools become more sophisticated and knowledge bases grow larger, the complexity of the knowledge engineering task increases. One approach to managing this complexity is to implement some form of apprenticeship learning program that is available to the expert (e.g. Wilkins, in press).

Providing process guidance in Aquinas

A subsystem called the dialog manager contains pragmatic heuristics to guide the expert through knowledge acquisition using Aquinas. Its help is important in the use of Aquinas, given the complexity of the Aquinas environment and the many elicitation and analysis methods available to the expert. The dialog manager makes decisions about general classes of actions and then recommends one or more specific actions providing comments and explanation if desired. This knowledge is contained in rules within the dialog manager in Aquinas. A session history is recorded so that temporal reasoning and learning may be performed (Kitto & Boose, in press).

Using Aquinas: building a programming language advisor

Aquinas is written in Interslip and runs on the Xerox family of Lisp machines. Subsets of Aquinas also run in an Interslip version on the DEC Vax and a "C/UNIX"-based portable version. The Aquinas screen is divided into a typescript window, map windows showing hierarchies, rating grid windows, and analysis windows (Fig. 8). Experts interact with Aquinas by text entry or by mouse through pop-up menus.

Following are the steps in a Aquinas session in which an expert is building a Programming Language Advisor. Novice software engineers and project managers would use such a system to help select programming languages for application projects. Aquinas guides the expert in putting knowledge into Aquinas's knowledge base, and continues through the making of a knowledge base for the S.1 expert-system shell. These steps are:

(1) ELICIT CASES AND THE INITIAL GRID (SOLUTIONS, TRAITS AND RATINGS)

The expert is first asked to specify the behavior of Aquinas's dialog manager. Then the expert enters several problem test cases and selects one for analysis. The *knowledge-acquisition language* case is selected (satellite tracking, accounting and government transaction cases are also entered). The cases are added to the case hierarchy and appear in the *map window* (Figure 8; upper right corner). Eventually experts may be able to select and modify grids and cases from a library; we expect that in several years this library will contain hundreds of hierarchies of grids.

The expert chooses to think about a language for developing a knowledge acquisition testbed, and enters potential candidates (Fig. 9). After five languages are entered, Aquinas adds an *ideal language* for this problem. This would be an ideal solution for the knowledge-acquisition case. The languages are added to the solution hierarchy as children. Then Aquinas asks the expert to enter traits based on differences and similarities between languages. This is the heart of Kelly's interviewing methodology. Aquinas uses it in several different ways as knowledge is expanded through elicitation and analysis.

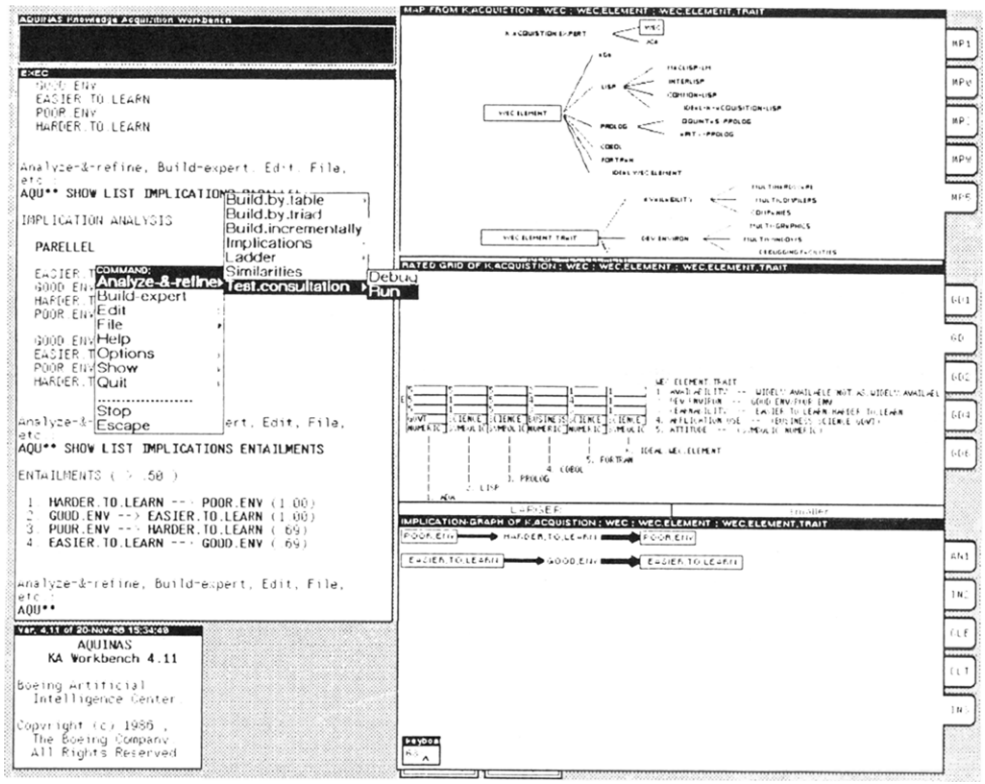

FIG. 8. Aquinas screen showing developing hierarchies for expert WEC, a rating grid, and an implication analysis graph of the grid.

Aquinas initially assumes that traits will be bipolar with ordinal ratings between 1 and 5. The expert is asked to rate each solution with regard to each trait, but the expert may specify different rating scales (unordered, interval, or ratio). Aquinas later assists in recognizing and changing types of rating scales.

Aquinas also elicits information about the importance of each trait. This knowledge is used later in the decision-making process.

(2) ANALYSE AND EXPAND THE INITIAL, SINGLE GRID

Once a grid is complete, an analysis is performed to show *implications* between various values of traits (see the lower right-hand window in Fig. 8). Implications are read from left to right, and the thickness of the arc shows the strength of the implication (HARDER TO LEARN implies POOR DEVELOPMENT ENVIRONMENT). A method similar to ENTAIL (Gaines & Shaw, 1985; Shaw & Gaines, in press) derives implications. Rating grid entries are used as a sample set and fuzzy-set logic is applied to discover inductive implications between the values. This method uncovers higher-order relationships among traits and later helps build trait hierarchies. The expert can also use an interactive process (implication review) to analyse and debug this information; the expert may agree or disagree with each implication. If the expert disagrees, the knowledge that led to the implication is

```
--- ELICITING ELEMENTS ---
Please enter a list of LANGUAGE elements for K-ACQUISITION, one to a line. When you're
done, enter a RETURN. Try to include at least one LANGUAGE that would NOT be good for K-
ACQUISITION (a counter example)
AQU** ADA
AQU** LISP
AQU** PROLOG
AQU** COBOL
AQU** FORTRAN
AQU** (CR)
Can you imagine a new ideal LANGUAGE for K-ACQUISITION?
AQU** YES
For now, it will be called IDEAL-K-ACQUISITION-LANGUAGE

--- BUILDING TRAITS BY TRIADS ---
When answering the following questions, remember to keep the K-ACQUISITION case in
mind

Think of an important attribute that two of ADA, LISP, and PROLOG share, but that the other
one does not. What is that attribute?
AQU** SYMBOLIC
What is that attribute's opposite as it applies in this case?
AQU** NUMERIC
What is the name of a scale or concept that describes
SYMBOLIC/NUMERIC?
AQU** ATTITUDE

Think of an important trait that two of LISP, PROLOG, and COBOL share, but that the other
one does not. What is that trait?
AQU** WIDELY AVAILABLE
What is that trait's opposite as it applies in this case?
AQU** NOT AS WIDELY AVAILABLE
What is the name of a scale or concept that describes WIDELY-AVAILABLE/NOT-AS-WIDELY-
AVAILABLE?
AQU** AVAILABILITY
      :

--- FILLING IN RATINGS ---
Please rate these things on a scale of 5 to 1, where 5 means more like SYMBOLIC and 1 means
more like NUMERIC. If neither one seems to apply, enter N(either). If both seem to apply, enter a
B(oth). If you would like to change the range or type of scale, enter C(hange scale)
SYMBOLIC(5) NUMERIC(1)
 ADA ** 5
 LISP ** 1
 PROLOG ** 1
 COBOL ** 5
 FORTRAN ** 5
 IDEAL-K-ACQUISITION-LANGUAGE ** 1
      :
```

FIG. 9. Aquinas asks the expert for an initial set of potential solutions to the first problem case. Then, the solutions are presented in groups of three, and the expert gives discriminating traits. Ratings are entered for each solution for each trait.

reviewed, and the expert can change the knowledge or add exceptions that disprove the implication (Boose, 1986). Certain types of *implication patterns* are also uncovered. Discovery of *ambiguous* patterns, for example, may mean that traits are being used inconsistently (Hinkle, 1965; Boose, 1986).

After the initial grid is complete, the dialog manager suggests a method to help the expert expand the grid. The method depends on the size of the grid, analysis of information in the grid, session history, and so on. The dialog manager inserts the appropriate command on the screen. The expert may change this recommendation or accept it by entering RETURN.

(3) TEST THE KNOWLEDGE IN THE SINGLE GRID

The dialog manager next recommends that the grid knowledge be tested by running a consultation. The expert is asked to provide desirable values for the traits

associated with an instance of the case under consideration. These values may be appended with a certainty factor and/or the tag ABSOLUTE to show an absolute constraint. Consultation questions are ordered according to a computed benefit/cost ratio that depends on both the generated system (e.g. entropy of a given trait, Quinlan, 1983) and the specified expert (e.g. cost of obtaining information) parameters. The questions may also be ordered according to an arbitrary specification given by the expert. Performance is measured by comparison of experts' expectations with Aquinas consultation results.

Two methods are available in Aquinas for turning rating values in grids into solution recommendations. One approach for turning rating values in grids into solution recommendations involves mapping this information onto certainty factor scales. Each rating in the grid is assigned a certainty factor weight based on its *relative strength* (a 5 is stronger than a 4), the *relative weight* the expert has assigned to the trait, and any *absolute constraints* that the expert has specified for the trait. In the test consultation, EMYCIN's certainty factor combination method (Adams, 1985) is used to combine the certainty factors. The result is a rank-ordered list of solutions with certainty-factor assignments. These certainty factors are also used when rules are generated for expert-system shells.

Another approach available employs Saaty's Analytic Hierarchy Process to order a set of possible solutions. Grid information obtained through pairwise comparisons or through regular rating grid methods is mapped onto *judgment matrices.* The *principal eigenvector* is computed for each matrix; the eigenvectors are normalized and combined to yield a final ranking of the solutions. Each solution has a score between 0·0 and 1·0. In a knowledge base consisting of multiple grids, these values are propagated through the hierarchies.

(4) BUILD HIERARCHIES (STRUCTURED AS SOLUTIONS AND TRAITS IN MULTIPLE GRIDS) FROM THE FIRST GRID

Next, the dialog manager recommends that the expert expand the trait and solution hierarchies by performing a *cluster analysis* (Fig. 10). Aquinas uses a method of single-link hierarchical cluster analysis based on FOCUS (Shaw & Gaines, in press *a*) to group sets of related solutions or traits. The junctions in the clusters can be seen as conjectures about possible new classes of solutions or traits. These more general trait or solution classes may be named and added to the hierarchies.

Laddering is also used to find traits at varying levels of abstraction (Boose, 1986). "Why?" questions are used to find more general traits;

> What is a new trait that says why you think GOOD-DEVELOPMENT-ENVIRONMENT should be true of a LANGUAGE for K-ACQUISITION?
> AQU** FASTER SYSTEM DEVELOPMENT.

"How?" questions help find more specific traits:

> How could a language for K-ACQUISITION be characterized by WIDELY-AVAILABLE?
> AQU** RUNS ON MULTIPLE HARDWARE
> AQU** MANY COMPILERS AVAILABLE
> AQU** MANY COMPANIES OFFER.

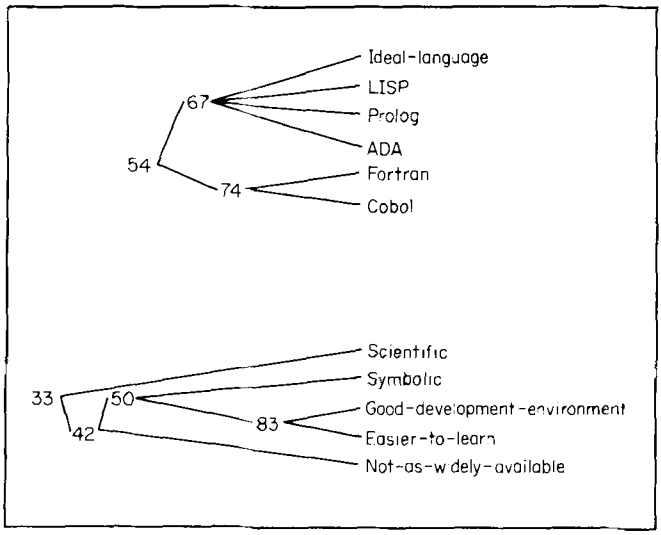

FIG. 10. Solution and trait clusters are formed from information in rating grids. The expert is asked to label nodes and expand clusters where possible; new traits are used to expand the hierarchies.

Experts stop expanding the trait hierarchies when they are able to provide direct grid ratings at these more specific trait levels. Ratings need not be explicitly given at each level of the trait and solution hierarchies, but can often be inferred from other grids in the knowledge base (e.g. induction from more specific examples or inheritance from more general ones) (Lieberman, 1986).

(5) USE SEVERAL RATING VALUE TYPES (TRANSFORM ORDINAL RATINGS TO NOMINAL AND INTERVAL RATINGS) TO REPRESENT KNOWLEDGE

Aquinas helps the expert convert a trait with ordinal values (DELIVERY-COST: HIGH-COST(5)/LOW-COST(1)) into a trait with ratio scaled rating values (DELIVERY-COST: (1500–60 000) DOLLARS-US). The expert re-rates the solutions in terms of the new values and these values appear on the grid Aquinas provides several forms of estimation help. Four estimation procedures are provided: START-&-MODIFY, EXTREME-VALUES, DECOMPOSITION, and RECOMPOSITION (Beyth-Marom & Dekel, 1985). In this instance, the EXTREME-VALUES procedure first asks for the least and greatest DELIVERY-COST one could imagine for the type of Lisp being considered. Through a series of questions, Aquinas helps shrink this range until a satisfactory estimate is given.

Aquinas also helps the expert change trait scale types by checking values associated with particular kinds of traits. For instance, bipolar traits that receive only extreme ratings (e.g. RUNS ON VAX/RUNS ON IBM) may be better represented with an unordered trait (e.g. COMPUTER TYPE).

(6) TEST KNOWLEDGE IN HIERARCHIES; TEST KNOWLEDGE FROM MULTIPLE EXPERTS

Another expert adds knowledge about programming language selection to the knowledge base and tests it. In the first consultation (Fig. 11), the user is interested

```
--- TEST CONSULTATION --
Would you like to run an EXISTING or NEW consultation?
AQU** EXISTING
What is the name of this existing consultation?
AQU** LISP-PROLOG-ADA-ONLY
This test consultation is named K-ACQUISITION LISP-PROLOG-ADA-ONLY.

Which K-ACQUISITION alternatives you would like to consider in this consultation (LISP-PROLOG-
ADA-ONLY). Enter them one to a line. If you wish all solutions to be considered, type ALL. When
done, press RETURN.
AQU** MACLISP-LM
AQU** INTERLISP
AQU** COMMON-LISP
AQU** QUINTUS-PROLOG
AQU** ADA-1
AQU** (CR)
The following experts know about MACLISP-LM, INTERLISP, COMMON-LISP, QUINTUS-PROLOG,
and ADA-1: WEC JCA. Would you like to exclude or weight any of these experts?
AQU** NO

Please indicate the desired trait selection values for LISP-PROLOG-ADA-ONLY solutions. Press
RETURN to indicate agreement with the default values, or type in a new value. Values may be
appended with a certainty factor in the form ' 8' and/or the word ABSOLUTE to indicate that the
value is an absolute constraint when selecting a type of LANGUAGE for K-ACQUISITION
(WIDELY-AVAILABLE(5), 1.0)** (CR)
(GOOD-DEVELOPMENT-ENVIRONMENT(5), 1.0)** (CR)
(LOW-COST(<45000 DOLLARS-US), 1.0, ABSOLUTE) (NOTE: THIS INCLUDES HARDWARE FOR A
WORKSTATION) ** <30000 DOLLARS-US 1.0 ABSOLUTE
      :
```

FIG. 11. The expert tests the knowledge by running a consultation. The expertise of two experts is used and consensus and dissenting solutions are given (see Fig. 12).

in selecting a particular version of Lisp, Prolog, or ADA for a knowledge acquisition project. Because of the many potential solutions, the user is given the opportunity to specify a subset for consideration. The solutions in this subset are called *solution candidates.*

Aquinas then asks for a set of absolute and preferred trait values for this consultation. The user enters an absolute constraint that only languages with a delivery cost of less than $30 000 will be considered. Patterns of constraints may be entered by using key words such as AND and OR. The user may accept default values entered in a previous consultation by pressing the RETURN key. If a default value has not been previously specified and the user types RETURN, the trait will be ignored in the inference process for this consulatation. The user's preference for

HARDWARE type is partitioned among three manufacturers by pairwise comparison (Fig. 12), which generates a ratio scaled set of preferences (Saaty, 1980).

The results of the consultation are presented to the user. For each solution, the *consensus* recommendation of the experts consulted is presented, followed by the weight of each expert that contributed to the recommendation. With multiple experts, it may sometimes be useful to examine a set of recommendations from a *dissenting* expert or group of experts. Since WEC's recommendations differed most from the consensus, these are listed as a dissenting opinion.

A general model illustrating the inference propagation path was shown in Fig. 6. For each expert consulted and for each level in that expert's solution hierarchy, a

```
        :
(COMPANIES(VAX .33, IBM .33, ATT .33), 1.0)** PAIRWISE
Please compare these values of HARDWARE with regard to their importance in contributing to an
overall high score for a particular type of LANGUAGE for <-ACQUISITION in the context of LISP-
PROLOG-ADA-ONLY.
Please compare VAX and IBM. Enter.
VAX = IBM if VAX and IBM are equally important
VAX>IBM or VAX<IBM if one of the pair is weakly more important
VAX>>IBM or VAX<<IBM if one is strongly more important
VAX>>>IBM or VAX<<<IBM if one is demonstrably or very strongly more important
VAX>>>>IBM or VAX<<<<IBM if one is absolutely more important
AQU** VAX<IBM

Please compare VAX and ATT. Enter:
VAX = ATT if VAX and ATT are equally important
VAX>ATT or VAX<ATT if one of the pair is weakly more important
VAX>>ATT or VAX<<ATT if one is strongly more important
VAX>>>ATT or VAX<<<ATT if one is demonstrably or very strongly more important
VAX>>>>ATT or VAX<<<<ATT if one is absolutely more important
AQU** VAX>>>>ATT
        :

Results for test consultation K-ACQUISITION LISP-PROLOG-ADA-ONLY:
 1: INTERLISP (.47 (WEC .5, JCA .5))
 2: QUINTUS-PROLOG (.40. (WEC 1.0))

Would you like to see the dissenting opinion for this consultation?
AQU** YES

The following dissenting opinion was given by WEC
Overall agreement with consensus .79
 1: QUINTUS PROLOG (.40)
 2: INTERLISP (.39)
```

FIG. 12. Test consultation (continued). The expert specifies "run-time" values for traits, entering an absolute cost constraint, and performing a pairwise comparison task to derive relative values for hardware. Consensus and dissenting opinions are given along with the weighted contributions of each expert.

partial problem model is constructed, evaluated, and abstracted in a bottom–up fashion through the trait hierarchy of that solution level. Through this process the solution is refined as the children of the best solutions are chosen for continued evaluation. Bottom–up abstraction takes place again in the trait hierarchy at the new solution level, and the cycle continues until all remaining solution candidates have been evaluated. Then an ordered list of solution candidates is obtained and combined with the results from other experts. This information from a single case may then be combined, if desired, with information from other cases to derive a final ranking of solution candidates. Users may override this general model of inference propagation by specifying explicit inference paths and parameters.

(7) EDIT, ANALYSE AND REFINE THE KNOWLEDGE BASE, BUILDING NEW CASES

Once the experts have entered information about one case, they describe additional cases. They could start from scratch by entering a list of relevant solutions and traits, but that would be inefficient if there were significant overlap in those required by a previously entered case and a new one (Mittal, Bobrow & Kahn, 1986). Aquinas allows an expert to copy pieces of hierarchies (and, optionally, their associated values) between cases. Information copied in this way can be modified to fit the new context. This facility may also be used to copy pieces of hierarchies between experts.

(8) FURTHER EXPAND AND REFINE THE KNOWLEDGE BASE

Hiearachies and rating grids continue to be used during the session to expand and refine the knowledge base. Work in progress is shown in Fig. 8. Aquinas contains a variety of other tools to help analyse and expand the knowledge base.

Comparison of experts (sources)

The MINUS tool (Shaw & Gaines, 1986) compares grids from different experts on the same subject and points our differences and similarities. This information has been used to manage structured negotiation between experts (Boose, 1986). SOCIOGRIDS features (Shaw & Gaines, in press *a*) will be available in the future to display *networks* of expertise. Nodes and relations in these networks show the degree of subsumption of one expert's grid over grids from other experts.

Incremental interviewing

Aquinas can use an incremental dialog to elicit new traits and solutions, one at a time, from the expert (Boose, 1986). This is useful when the expert does not have a list of solutions to start a grid and in other situations during knowledge refinement.

Trait value examination

New solutions can be identified by asking the expert to "fill in holes" in the values of trait ranges. For instance, no solution may exist with a rating of 2 on some ordinal trait scale; the expert is asked if one can be identified:

> What is a new LANGUAGE that would receive a value of 2 on the scale SCIENTIFIC(5)/BUSINESS(1)?

New traits can also be identified by forming triads based on ratings: if LISP and

PROLOG are rated 5 on SCIENTIFIC(5)/BUSINESS(1), and ADA is rated 4, the expert is asked:

> What is a new trait having to do with SCIENTIFIC/BUSINESS that makes LISP and PROLOG similar yet different from ADA?

Trait range boundary examination
Important traits can frequently be identified by exploring the boundaries of trait ranges:

> You said that the range of DELIVERY-COST for LISP for the K-ACQUISITION case was 1500 to 60 000 DOLLARS-US. Can you think of any conditions in the future that might make DELIVERY-COST LESS THAN 1500?
> AQU** YES
> Enter conditions in terms of traits , one to a line; enter a RETURN when done.
> AQU** HARDWARE BREAKTHROUGH-LISP ON A CHIP
> AQU** (CR)
>
> Can you think of any conditions in the future that might make DELIVERY-COST GREATER THAN 60 000?
> AQU** YES
> Enter conditions in terms of traits, one to a line, enter a RETURN when done.
> AQU** VERY POWERFUL HARDWARE
> AQU** PARALLEL ARCHITECTURES AVAILABLE
> AQU** (CR)

Completeness checking
A single grid can be used as a table of examples. If the table is incomplete, the expert is asked to fill in other examples (Boose, 1986).

Combine similar traits
Sometimes different labels are used for the same underlying concept. This can be discovered when a similarity analysis is performed (functionally equivalent traits with different labels may be uncovered). If the expert cannot think of a new solution to separate identical traits, then the traits may be combined into a single trait.

(9) GENERATE RULES FOR EXPERT-SYSTEM SHELLS
The expert is the judge of when the point of diminishing returns has been reached within Aquinas. When such a point is reached, a knowledge base is generated for an expert-system shell, and development continues directly in that shell. Similarity and implication analyses allow experts to determine whether traits or solutions can be adequately and appropriately discriminated from one another. The system provides correlational methods for comparing the order of Aquinas recommendations with an expert's rankings.

Aquinas can generate knowledge bases for several expert-system shells (KEE, KS-300/EMYCIN, LOOPS, M.1, OPS5, S.1, and others). The knowledge contained in grids and hierarchies is converted within Aquinas into rules, and the rules are

formatted for a particular expert-system shell. Appropriate control knowledge is also generated when necessary. Rules are generated with screening clauses that partition the rules into subsets. An *expert clause* is used when expertise from multiple experts is weighted and combined together. A *case clause* controls the focus of the system during reasoning.

Four types of rules are generated:

(1) *Implication rules* are generated from arcs in the implication graph and conclude about particular traits. The conclusion's certainty factor is proportional to the strength of the implication. The use of implication rules restricts search and lessens the number of questions asked of users during consultations;

(2) *Solution rules* conclude about a particular solution class. The conclusion's certainty factor is derived from a combination of the *grid rating strength* and the *trait weight*;

(3) *Absolute rules* are generated when the expert places an absolute constraint on the value of a trait. Sometimes information about absolute constraints is included elsewhere when knowledge bases for expert system shells are generated;

(4) *Specialization/generalization rules* are derived from information in the hierarchies and are used to propagate hierarchical information.

Discussion

GENERAL ADVANTAGES AND DISADVANTAGES OF AQUINAS

Improved process efficiency and faster knowledge-base generation

Aquinas inherits the advantages of ETS: rapid prototyping and feasibility analysis, vocabulary, solution and trait elicitation, interactive testing and refinement during knowledge acquisition, implication discovery, conflict point identification, expert-system shell production, and generation of expert enthusiasm (Boose, 1986). It is much easier for users to learn knowledge-based system concepts through using Aquinas than through reading books or attending classes (i.e. rules are automatically generated and used dynamically in consultations; new vocabulary is incrementally introduced).

ETS, still in use at Boeing, has been employed to build hundreds of single-grid prototype systems. Single grids as large as 42×38 (1596 ratings) have been built. Alternative approaches may be tested with little expenditure of resources. Knowledge bases have been generated for expert-system shells that contain over 2000 rules. Typically, something on the order of a 15×10 grid is built that generates several hundred rules.

Over 30 prototype systems have been built during the development of Aquinas (an AI book consultant, and AI tool advisor, a course evaluation system, a customer needs advisor, a database management system consultant, an investment advisor, a management motivation analyser, a personal computer advisor, a personality disorder advisor, a product design and impact advisor, a robotic tool selector, a Seattle travel agent, and a wine advisor, among others). Some of these systems contain thousands of ratings arranged in hierarchical grids. The Programming Language Advisor session took less than 2 h with each of the two experts.

Improved knowledge-base quality

Aquinas offers a rich knowledge representation and reasoning environment. We believe that Aquinas can be used to acquire knowledge for significant portions of most structured selection expert-system problems. Hierarchies help the expert break down problems into component parts and allow reasoning at different levels of abstraction. Varying levels of precision are specified, with multiple types of rating scales when needed. Multiple methods for handling combining uncertain information are available based on needed precision and convenience. Knowledge from multiple experts may be combined using Aquinas. Users may receive dissenting as well as consensus opinions from groups of experts, thus getting a full *range* of possible solutions. Disagreement between the consensus and the dissenting opinion can be measured to derive a *degree of conflict* for a particular consultation. The system can be used for cost-effective group data gathering (Boose, in press).

Analytic tools help uncover inconsistencies and circularities in the growing knowledge base.

Better knowledge-base maintenance and comprehensibility

Elicitation, structuring, analysis, and testing of knowledge is based on specific cases. When knowledge in Aquinas is updated, it is done so with respect to a specific case. Addition of new knowledge in this way can be strictly controlled by the expert; the tendency for local changes to degrade other cases is thus curbed.

The expert builds and refines knowledge in rating grids and hierarchies—not directly in production rules. As a result, knowledge at this higher level of abstraction is more compact, comprehensible, and easier to maintain.

The growing collection of rating grids and case knowledge represents an important resource for building a variety of knowledge-based systems. Knowledge is stored explicitly with associated problem cases, making knowledge bases easier to update and maintain.

Currently, a user may copy and change any portion of the Aquinas knowledge base during a consultation. In the future, each expert will be able to protect areas of knowledge. The expert may believe protection is necessary because some knowledge should not be changed or because the knowledge has commercial value.

Extensions to personal construct theory methods

Aquinas significantly extends existing personal construct theory methods. Rating grid knowledge can be tested and used interactively to make decisions; rating grid information may be arranged and coupled in hierarchies; multiple rating scale types are available (not just bipolar ordinal scales); many grid analysis tools are available in a single workbench.

Process complexity

Aquinas is not as easy to use as was ETS using single grids. There are many elicitation and analysis tools for a novice to understand; the decision-making process and inference engine can be set up to work in several different ways. We expect that continuing improvements in the dialog manager will help make the system more comprehensible and decrease the learning time for new users.

THEORETICAL ISSUES—KNOWLEDGE ELICITATION

Personal construct psychology methods provide no guarantee that a *sufficient* set of knowledge will be found to solve a given problem. Aquinas attempts to expand the initial subset of solutions and traits based on problem-solving knowledge for specific cases. The goal is to solve enough cases to that the knowledge is sufficient to solve *new* cases. This is the methodology of knowledge engineering in general; Aquinas helps make the process explicit and manageable.

Hierarchical decomposition can be used to build intuitive, comprehensible models that seem to behave in reasonable ways. One disadvantage is that some problems do not easily fit the hierarchical model. It also may be true that a particular problem would best be represented by a *collection of conflicting hierarchies* (hierarchies for mechanical problems tend to model structure *or* function, not both, and both may be necessary).

The use of multiple rating value types provides more flexibility, convenience, and precision in representing knowledge. However, deciding a particular type of variable to use can be a complex task. The dialog manager offers some assistance, but the expert usually must learn appropriate usage of rating types through experience.

Experts develop Aquinas knowledge bases serially. In the future, we would like to build a participant system in which many experts could dynamically share rating grids and hierarchies (Chang, 1985).

ANALYSIS AND INFERENCE

Multiple analysis tools and elicitation methods in Aquinas help the expert think about the problem in new ways and tend to point conflicts and inconsistencies over time. Lenat (1983*a*, *b*) argues that knowledge representations should shift as different needs arise. This should lead to a better problem and solution descriptions, and, in turn, to better problem solving.

Inference in Aquinas is efficient because the problem space is partitioned. Information in the trait hierarchies is attached to particular levels of solutions. Although no formal studies have been conducted, consultation results using the methods described above seem reasonable.

Rule generation for expert-system shells is straightforward. Development of the knowledge base can continue in an expert system shell that may offer advantages of speed, specialized development and debugging facilities, and inexpensive hardware.

FUTURE DIRECTIONS

We intend to build a knowledge-acquisition environment that includes specific domain knowledge for specialized application areas and can acquire knowledge for synthetic problems, combining features from other knowledge acquisition tools such as MDIS (Antonelli, 1983), DSPL (Brown, 1984), MORE (Kahn, Nowlan & McDermott, 1985), MOLE (Eshelman, Ehert, McDermott & Tan, in press), and TKAW (Kahn, Breaux, Joeseph & De Klerk, in press).

Presently Aquinas works best on those problems whose solutions can be comfortably enumerated (*analytic* or *structured selection* problems such as classification or diagnosis) as opposed to problems whose solutions are built up from components (*synthetic* or *constructive* problems such as configuration or planning).

Simple classification can be thought of as a single-decision problem (handled by ETS). Complex structured selection problems may require a set of linked data abstraction/solution refinement decisions (Aquinas). The next step may be to generalize this process to acquire and represent knowledge for planning, configuration, and design problems where the order of linked decisions in solution hierarchies may represent precedence of events or goals rather than just solution refinement. In these problems hierarchies may be assembled at consultation time rather than constructed totally in advance as they are currently. Grid cells might sometimes contain an arbitrary computation rather than a rating. These would include results of functions (such as found in spreadsheets) or data base retrievals. Deeper models of the structure and function of physical systems could be modeled.

An important step in expanding the knowledge-acquisition workbench concept is the linking together of other specialized tools. At the Boeing Knowledge Systems Laboratory we are investigating ways of integrating diverse knowledge representations from different laboratory projects so that this may be more easily accomplished. In the domain of knowledge acquisition, we feel that the approach used in SALT (Marcus, McDermott & Wang, 1985; Marcus & McDermott, in press; Marcus, in press) is particularly promising. SALT is a system that interviews experts to build knowledge bases for certain types of constructive problems (its first use was to configure elevators). We are also interested in generating knowledge sources for BBB, a blackboard system that has been successfully applied to a variety of problems (Benda, Baum, Dodhiawala & Jagannathan, 1986).

Development of the Aquinas workbench will continue in an incremental fashion. Techniques will be continuously integrated and refined to build an increasingly more effective knowledge-acquisition environment.

Thanks to Roger Beeman, Miroslav Benda, Kathleen Bradshaw, William Clancey, Brian Gaines, Cathy Kitto, Ted Kitzmiller, Art Nagai, Doug Schuler, Mildred Shaw, David Shema, Lisle Tinglof-Boose, and Bruce Wilson for their contributions and support. Aquinas was developed at the Knowledge Systems Laboratory, Advanced Technology Center, Boeing Computer Services in Seattle, Washington.

References

ADAMS, J. (1985). Probabilistic reasoning and certainty factors. In BUCHANAN, B. & SHORTLIFFE, E., Eds, *Rule-Based Expert Systems. the MYCIN Experiments of the Standford Heuristic Programming Project,* Reading. Massachusetts: Addison–Wesley.

ALPERT, M. & RAIFFA, H. (1982). A progress report on the training of probability assessors. In KAHNEMAN, D., SLOVIC, P. & TVERSKY, A. Eds, *Judgment under Uncertainty: Heuristics and Biases,* New York: Cambridge University Press.,

ANTONELLI, D. (1983). The application of artificial intelligence to a maintenance and diagnostic information system (MDIS). *Proceedings of the Joint Services Workshop on Artificial Intelligence in Maintenance,* Boulder, Colorado.

BENDA, M., BAUM, L. S., DODHIAWALA, R. T. & JAGANNATHAN, V. (1986). Boeing blackboard system. *Proceedings of the High-Level Tools Workshop,* Ohio State University, October 1986.

BENDA, M., JAGANNATHAN, V. & DODHIAWALA, R., On optimal cooperation of knowledge sources. *Workshop on Distributed Artificial Intelligence,* Gloucester, Massachusetts.

BEYTH-MAROM, R. & DEKEL, S. (1985). *An Elementary Approach to Thinking under Uncertainty.* London: Lawrence Erlbaum Associates.

BOOSE, J. H. (1984). Personal construct theory and the transfer of human expertise. *Proceedings of the National Conference on Artificial Intelligence,* Austin, Texas.

BOOSE, J. H. (1985). A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man–Machine Studies,* **23,** 495–525.

BOOSE, J. H. (1986). *Expertise Transfer for Expert System Design.* New York: Elsevier.

BOOSE, J. H. (1987). Rapid acquisition and combination of knowledge from multiple experts in the same domain. *Future Computing Systems Journal* In press.

BROWN, D. E. (1984). Expert systems for design problem-solving using design refinement with plan selection and redesign. Ph.D. Thesis, Ohio State University, CIS Department, Columbus, Ohio.

BYLANDER, T. & CHANDRASEKARAN, B. (1987). Generic tasks in knowledge-based reasoning: the "Right," level of abstraction for knowledge acquisition. *International Journal of Man–Machine Studies.* In press.

BYLANDER, T. & MITTAL, S. (1986). CSRL: a language for classificatory problem solving and uncertainty handling. *AI Magazine,* August.

CHANDRASEKARAN, B. (1986). Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. *IEEE Expert,* Fall.

CHEESEMAN, P. (1985). In defense of probability. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, California.

CLANCEY, W. (1986). Heuristic classification In KOWALIK, J. Ed., *Knowledge-Based Problem-Solving.* New York: Prentice–Hall.

COOMBS, C. H., DAWES, R. M. & TVERSKY, A. (1970). *Mathematical Psychology.* Englewood Cliffs, New Jersey: Prentice–Hall.

ESHELMAN, L. EHRET, D., MCDERMOTT, J. & TAN, M. (1987). MOLE: a tenacious knowledge-acquisition tool *International Journal of Man–Machine Studies.* In press.

GAINES, B. R. (1987). An overview of knowledge acquisition and transfer. *International Journal of Man–Machine Studies.* In press.

GAINES, B. R. & SHAW, M. L. G. (1987). New directions in the analysis and interactive elicitation of personal construct systems. In SHAW, M. L. G. Ed., *Recent Advances in Personal Construct Technology.* New York: Academic Press.

GAINES, B. R. & SHAW, M. L. G. (1985). Induction of inference rules for expert systems. *Fuzzy Sets and Systems.*

GORDON, J. & SHORTLIFFE, E. (1985). The Dempster–Shafer theory of evidence. In BUCHANAN, B. & SHORTLIFFE, E. Eds, *Rule-Based Expert Systems: the MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, Masachusetts: Addison–Wesley.

GRUBER, T. & COHEN, P. (1987). Design for acquisition principles of knowledge system design to facilitate knowledge acquisition. *International Journal of Man–Machine Studies.* In press.

HENRION, M. (1986). Propagating uncertainty by logic sampling in Bayes' networks. *Proceedings of the Second Workshop on Uncertainty in Artificial Intelligence,* Philadelphia, Pennsylvania.

HINKLE, D. N. (1965). The change of personal constructs from the viewpoint of a theory of implications. *Ph. D. Thesis,* Ohio State University, Ohio.

HOWARD, R. A. & MATHESON, J. E. (1984). Influence diagrams. In HOWARD, R. A. & MATHESON, J. E. Eds, *Readings on the Principles and Applications of Decision Analysis,* Menlo Park, California: Strategic Decisions Group.

HURLEY, R. (1983). *Decisions Tables in Software Engineering,* New York: Van Nostrand Reinhold.

KAHN, G., NOWLAN, S. & MCDERMOTT, J. (1985). MORE: an intelligent knowledge acquisition tool. *Proceedings of the Ninth Joint Conference on Artificial Intelligence,* Los Angeles, California.

KAHN, G. S., BREAUX, E. H., JOESEPH, R. L. & DEKLERK, P. (1987). An intelligent mixed-initiative workbench for knowledge acquisition. *International Journal of Man–Machine Studies.* In press.

KELLY, G. A. (1955). *The Psychology of Personal Constructs,* New York: Norton.

KITTO, C. & BOOSE, J. H. (1987). Heuristics for expertise transfer: the automatic management of complex knowledge-acquisition dialogs. *International Journal of Man–Machine Studies.* in press.

LENAT, D. (1983*a*). The nature of heuristics. *Artificial Intelligence,* **19,**

LENAT, D. (1983*b*). The nature of heuristics. *Artificial Intelligence,* **21.**

LENAT, D., PRAKASH, M. & SHEPARD, M. (1986). CYC: using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine,* **6.**

LIEBERMAN, H. (1986). Using prototypical objects to implement shared behavior in object-oriented systems. *Proceedings of the Object-Oriented Programming Systems, Languages, and Applications Workshop,* Portland, Oregon

MARCUS, S., MCDERMOTT, J. & WANG, T. (1985). Knowledge acquisition for constructive systems. *Proceedings of the Ninth Joint Conference on Artificial Intelligence,* Los Angeles, California.

MARCUS, S. & MCDERMOTT, J. (1987). SALT: a knowledge acquisition tool for propose-and-revise systems. *Carnegie–Mellon University Department of Computer Science technical report.* In press.

MARCUS, S. (1987). Taking backtracking with a grain of SALT. *International Journal of Man–Machine Studies.* In press.

MCDERMOTT, J. (1986). Making expert systems explicit. *Proceedings of the IFIP Congress,* Dublin, Ireland.

MICHALSKI, R. S. (1978). *Designing Extended Entry Decision Tables and Optimal Decision Trees Using Decision Diagrams.* Urbana, Illinois: Intelligent Systems Group, Artificial Intelligence Laboratory, Department of Computer Science, University of Illinois.

MICHALSKI, R. S. & WINSTON, P. (1985). Variable precision logic. *MIT AI Memo,* Artificial Intelligence Laboratory, Masachusetts Institute of Technology, 857.

MICHALSKI, R. S., *Machine Learning.* Plenary talk at the *AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Canada, November, 1986.

MITTAL, S., BOBROW, D. & KAHN, K. (1986). Virtual copies: at the boundary between classes and instances. *Proceedings of the Object-Oriented Programming Systems, Languages, and Applications Workshop,* Portland, Oregon.

PEARL, J. (1986). Fusion, propagation and structuring in belief networks. *Technical report CSD-850022, R-42-VI-12,* Cognitive systems Laboratory, Computer Science Department, University of California, Los Angeles, California.

QUINLAN, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In MICHALSKI, R. S., CARBONELL, J. G. & MITCHELL, T. M. Eds, *Machine Learning—An Artificial Intelligence Approach,* Vol. 1. Tioga: Palo Alto, California.

REBOH, R. & RISCH, T. (1986). SYNTEL: knowledge programming using functional representations. *Proceedings of National Conference on Artificial Intelligence (AAAI-86),* Philadelphia, Pennsylvania.

REBOH, R., RISCH, T., HART, P. E. & DUDA, R. O. (1986). Task-specific knowledge representation: a case study. *Proceedings of the High-Level Tools Workshop,* Ohio State University.

SAATY, T. L. (1980). *The Analytic Hierarchy Process.* New York: McGraw–Hill.

SHAFER, G. (1976). *A Mathematical Theory of Evidence.* Princeton: University Press.

SHAFER, G. & TVERSKY, A. (1985). Languages and designs for probability judgment. *Cognitive Science,* **9,** 309–339.

SHASTRI, L. & FELDMAN, J. (1985). Evidential reasoning in semantic networks: a formal theory. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, California.

SHAW, M. L. G. & GAINES, B. R. (1987*a*). PLANET: a computer-based system for personal learning, analysis, negotiation and elicitation techniques. In MANCUSO, J. C. & SHAW, M. L. G: Eds, *Cognition and Personal Structure: computer Access and Analysis.* Praeger Press. In press.

SHAW, M. L. G. & GAINES, B. R. (1987*b*). Techniques for knowledge acquisition and transfer. *International Journal of Man–Machine Studies.* In press.

SPETZLER, C. & STAL VON HOLSTEIN, C. (1983). Probability encoding in decision analysis. In HOWARD, R. & MATHESON, J. Eds, *Readings on the Principles and Applications of Decision Analysis,* Vol. 2. Palo Alto, California: Strategic Decisions Group.

SPIEGELHALTER, D. J. (1986). Probabilistic reasoning in predictive expert systems. In KANAL, L. N. & LEMMER, J. Eds, *Uncertainty in Artificial Intelligence.* Amsterdam: North-Holland.

SZOLOVITZ, P. & PAUKER, S. (1978). Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence,* **11,**

WALLSTEN, T. & BUDESCU, D. (1983). Encoding subject probabilities: a psychological and psychometric review. *Management Science,* **29.**

WILKINS, D. C. (1987) Knowledge base debugging using apprenticeship learning techniques. *International Journal of Man–Machine Studies.* In press.