FROM ETS TO AQUINAS: SIX YEARS OF KNOWLEDGE ACQUISITION TOOL DEVELOPMENT

John H. Boose, Jeffrey M. Bradshaw, Catherine M. Kitto, David B. Shema
Knowledge Systems Laboratory, Boeing Advanced Technology Center
Boeing Computer Services, P.O. Box 24346, Seattle, WA, 98124, USA

ABSTRACT

Two major forces brought about the development of the knowledge acquisition tools ETS (the Expertise Transfer System) and Aquinas: technology push and application pull. Ideas from many areas were gradually integrated to meet the growing demands of knowledge-based systems problems at The Boeing Company. This paper briefly traces the history of this development and the sources of many of our ideas, describes features and the reasons that they were added, and illustrates typical applications at each stage of evolution.

First a brief analysis of the forces that led to the development of our knowledge acquisition tools is presented. Then a history of tool growth is shown. Included are various features that were introduced and the types of help they provided. Typical applications at each stage of development are illustrated. Finally, we list the next hurdles to be overcome and show how we expect to overcome some of them in the next few years.

1. DYNAMIC TENSION: TECHNOLOGY PUSH AND APPLICATION PULL

At each phase in the growth of new tools, there should be a balance between technology push and application pull. Developers have ideas they feel might be useful for solving a future application problem - *technology push* - and applications present more immediate problems that need to be solved - *application pull*. Technology push tends to be farsighted and application pull tends to be shortsighted. When technology push is the only force, there is a danger that any tools developed will not be useful since they are not anchored in real problems, only a developer's vision of what the problems may be. When application pull is the only force, tools tend to be developed for special purposes and are hard to generalize to other problems. In such a demand-driven environment, revolutionary (or even evolutionary) breakthroughs usually do not occur. Ideally, both of these forces should act on a technology development project at the same time, leading to tools that are farsighted, general, and yet useful for a variety of problems.

Boeing's Advanced Technology Center provides a unique environment that fosters the interplay of technology push and application pull. The Center sponsored an Associates' Program that helped transfer artificial intelligence technology to the rest of Boeing. Associates spent one year at the Center, where they attended courses and developed prototypes to solve a specific problem in their home division. When the year's training was complete, Associates returned to their groups where they continued to develop and eventually field the applications. Over 95 Associates graduated from the program; the majority of them built knowledge-based systems. At the same time, the Center was working with universities to import advanced technology projects and was developing its own projects in the areas of vision, robotics, speech understanding, natural language, machine learning, and knowledge-based systems.

2. SEEDS: THE ARCHITECTURAL PROFILE SYSTEM

Our knowledge acquisition tools are rooted in Personal Construct Theory (Kelly, 1955) and related methods, especially techniques based on repertory grid elicitation and analysis. Boose first explored repertory grid techniques in 1976 while a student at the University of Maryland School of Architecture, working in the Computer Science Department's Artificial Intelligence group. He was trying to build a software system that would help people design their own homes. The first component, the Architectural Profile System, was supposed to find out how people felt about house environments in terms of textures, visual use of space, materials, utility, colors, overall design style (Cape Cod, Modern, etc.), privacy, use of glass, relationship to the outdoors, and so on. Once variables were identified, the system was to match the user's preferences to existing styles and canonical templates. Other components of the system would then help the user complete a detailed design.

Boose explored the psychometric literature and implemented a repertory grid elicitation program in Fortran on a Univac 1108, based on Kelly's description of the Role Repertory Grid Test. Kelly's original repertory grid was used in psychotherapy. Elements were people the client knew; triadic comparison of these people helped elicit important characteristics (constructs) that could show clients how they viewed themselves and others. In the Architectural Profile System, descriptions of environmental situations were used as elements (for example, "A place where you like to read," "A crowded place that you like," "A place where you feel lonely"). Triadic elicitation produced constructs about the built environment (for example, small space / large space, hard surfaces

/ soft surfaces, comfortable furniture / showy furniture). Kelly's original non parametric factor analysis was implemented to produce constellations of constructs; these constructs then fed a Lisp-based laddering process (Hinkle, 1965) in which questions about specialization and generalization were asked in an attempt to expand the constructs and link them into a network. Data were gathered from about 30 subjects who used the repertory grid elicitation program.

This was as far as the system progressed. The next steps would have been to gather data about different architectural styles and components based on the types of constructs that were elicited (some data were assembled from styles based on about 20 well known architects), build a starting set of templates indexed to these style data, link the laddered network to the index, and build a graphics interface to show floor plans, elevations, and descriptions retrieved while using the system.

3. KAQ: A FRONT END FOR TEIRESIAS IN EMYCIN

In the spring of 1983, Boose joined the Boeing Artificial Intelligence Center (part of the Advanced Technology Applications Division, later to become the Advanced Technology Center). The Associates' Program started that summer with about 12 students. Early problems faced by the Associates who tried to build expert systems provided application pull for the initial knowledge acquisition efforts at the Center.

The first knowledge-based system "shell" used was a version of EMYCIN with a graphic tracing interface that ran on the Xerox 1100 Lisp machine. This system, developed at Teknowledge, was called KS300. Part of TEIRESIAS (Davis and Lenat, 1982), an early knowledge transfer tool, was embedded in this shell.

TEIRESIAS in EMYCIN provided debugging help after an initial knowledge base had been entered. The system also provided an intelligent rule and attribute editor and an interviewing dialog to help the user construct an initial knowledge base. This combination provided an excellent rapid prototyping environment for structured selection problems that is still not equaled by most of today's knowledge-based system shells. Still, Associates using this environment faced problems that still challenge knowledge engineers:

1. How should the knowledge base be structured; in EMYCIN terms, how should the context tree be structured?

2. How should rules (or frames) be chained together to properly drive reasoning?

3. When uncertain knowledge is needed, how can certainty factors be controlled to improve knowledge base performance?

4. What are the variables (variously called attributes, traits, characteristics, parameters, concepts, and dimensions, or in EMYCIN, *PARMS*) that should be included in rules or frames? What are the potential values for these variables? Which ones are more important, and how does this affect the knowledge base structure and reasoning?

Since 1) EMYCIN helped build a simple starting context tree, 2) rule chaining in initial systems tends to be shallow, and 3) issues about uncertain knowledge tend to be uncovered later in the knowledge engineering process, identifying and using variables for rules were the first real problems an Associate faced.

Boose thought repertory grids might be effective in helping Associates identify variables for rules. If an initial conclusion or solution set could be provided, then triadic elicitation could be used to help an Associate's expert generate an initial list of variables and values. Once these variables were identified, they could be used to help build rules in KS300. This use of repertory grids and, later, related methods, provided much of the technology push in 1983 and 1984.

An attempt was made to resurrect the Architectural Profile System on the Center's VAX but the Fortran had a different "flavor" and many of the old Univac library modules had no corresponding modules on the VAX. In July, 1983, Boose started building a triadic elicitation system in Interlisp on the Xerox 1100 Lisp machine. Because of the advanced programming environment on this machine, a basic grid system was written in about a month. This system was called KAQ (for Knowledge AcQuisition; usually pronounced QUACK, or DUCK as it was dubbed by a colleague). KAQ was a substantial improvement over the Architectural Profile System. It included pop-up menus and windows, graphics with selectable objects, trait value *ranges* (from 1 to 5) as opposed to binary ratings, and allowed the recording of trait weights. A Wood Advisor was developed to test the system as it was being built; it used about 15 wood types and ten attributes. One of the Associates, Art Nagai, was building a database management system advisor using KS300 and had gotten stuck. This was the first "real" problem KAQ attacked. The system seemed to perform reasonably well and offered Nagai new insights that he applied to his advisor. A series of demonstrations was given to colleagues, Associates, and frequent visitors who toured the Center.[1]

---

[1] Frequent demonstrations were, and still are, a way of life at the Center. We estimate that we have given over 300 demonstrations of KAQ, ETS, NeoETS, and Aquinas in the past six years. These demonstrations and use of the tools by Associates and others provide valuable feedback for new applications, increased utility, system limitations to be overcome, and improvements to the user interface.

In September 1983, KAQ contained the following features (tools or features are listed with bullets; uses and advantages of the tool are shown in italics):

- Solution listing
  Identify potential solutions to the problem
- Triadic comparison of solutions
  Identify important discriminating traits (variables) for rules
- Ratings and trait weightings (1-5)
  Get judgments; show operational view of traits based on concrete solution objects (elements)
  Elicit relative importance of traits
- Graphic editors- labels, ratings, addition, deletion
  Easily change information associated with grid

At this point, an interesting collection of articles was found describing tools and methods for extending repertory grid analysis (Shaw, 1981). Boose began applying ideas from some of the tools, notably ENTAIL (Gaines and Shaw, 1981) and DYAD (Keen and Bell, 1981).

The article including ENTAIL sketched a method that used fuzzy logic for discovering *entailments* between values of the traits (we later came to call these entailments *implications*). These implications showed logical generalizations, using the grid data as a sample set. An algorithm was implemented based on ENTAIL that added graphic output to KAQ to show networks of implications. Experts using KAQ could agree or disagree with these generalizations. They were often surprised by implications that they subsequently realized were true. These could give users new insights about their problems by transforming the knowledge and changing the way it was viewed. This idea became an important aspect of future work as more analysis tools were added.

When experts disagreed with implications, methods were established for reviewing the ratings in the grid for errors, adding "exception" solutions to the grid to make the offending implications disappear, and using laddering to check the consistency of use of the traits across all the solutions (Boose, 1984, 1985, 1986a). These analyses were first carried out manually; later they were implemented in ETS. Implication generation and review were the first analysis tools for debugging the information in repertory grids. Implicit in the point of view that led to their development was that grid data would be used for problem-solving - a very different use than the one originally intended by Kelly.

Implications also proved useful for identifying knowledge at different levels of abstraction. They could, for example, point out "part of," "kind of," or subsumption relations. Implications could also show the relative independence of traits (the ENTAIL algorithm produced an entailment strength from 0.0 to 1.0). For some applications, it is important to make traits in rules as independent as possible; this technique provided a measure of trait independence.

Hinkle, one of Kelly's students, had used other interviewing methods to generate implications (Hinkle, 1965). He analyzed these implications for patterns that he termed parallel, reciprocal, orthogonal, and ambiguous. These patterns could point out inconsistency, ambiguity, and equivalence among constructs. This pattern analysis was implemented in KAQ (Boose, 1986a).

DYAD used an incremental interviewing method to develop a grid. This would be useful when an expert could not readily list a possible conclusion set for a problem, so it was also implemented in KAQ (Boose, 1986a). Another article mentioned the use of ratings N (for neither pole applies) and B (for both poles apply). Situations often arose in which both of these ratings would be useful, typically when an element was outside a construct's range of convenience, so this feature was also added to KAQ.

In summary, by early October 1983, the following features had been added to KAQ:

- Implication analysis (fuzzy set generalization)
  Show trait value relationships at higher level of abstraction
  Summarize grid information
  Note "problems" and fix manually (bad ratings, exceptions, consistency)
  Provide another view of data
  Measure trait independence
- Implication Analysis and Examination
  Agreement
  Is it obvious?
  Is it new to the expert?
  Disagreement
  Are ratings correct?
  Is there an exception?
  Are traits being used consistently?

- o Hinkle's implication analysis
  Point out relationships, especially problems (ambiguities and equivalences)
- o DYAD dialog (incremental interviewing)
  Expand grid with important solutions and traits
  Use if expert has no initial set of solutions
- o N (neither pole applies) and B (both poles apply) ratings
  Note when a trait does not apply comfortably to a particular solution

Did all of this help knowledge engineers? Art Nagai, for example, claimed that KAQ gave him significant help for advancing the Database Management System Advisor. Others who used KAQ said they received insights that helped them start prototyping or enhancing projects using KS300.

Other typical applications from this period include a jet engine diagnosis aid, a communication network bug analyzer, and a Seattle Japanese restaurant advisor. Most of the advantages and disadvantages of repertory grid methodology were uncovered in the next few months as KAQ was iteratively changed and improved. These are discussed below.

4. ETS: EMBEDDED REASONING AND INTEGRATED ANALYSIS

In October 1983 KAQ was renamed the Expertise Transfer System, ETS (paying homage to Davis, who coined the term "expertise transfer" in his thesis on TEIRESIAS).

Boose realized that KS300 rules could be generated automatically from implications. In an implication, a value or "pole" of a trait implies another value with a certain strength. By eliciting an additional piece of information from the expert (the "concept" or "scale" name for each trait), ETS could transform an implication directly into a KS300 rule:

Implication: MORE-EXPENSIVE implies FARTHER-AWAY (strength)

KS300 rule: If COST = MORE-EXPENSIVE
Then DISTANCE = FARTHER-AWAY (CF)

At first the fuzzy implication strength (from 0.0 to 1.0) was simply mapped directly onto KS300's certainty factor scale (from -1.0 to 1.0). Later, other methods (described below) were tried.

These rules pointed out relationships between traits, but another kind of rule, one that related values of traits to specific solutions, was needed. It turned out to be easy to generate these kinds of rules, too. For example, if the solution PARIS was rated 5 on the trait DISTANCE (CLOSER (1) / FARTHER-AWAY (5)), then the following rule could be generated:

If DISTANCE = FARTHER-AWAY
Then SOLUTION = PARIS (CF)

Another rule could also be generated:

If DISTANCE = CLOSER
Then SOLUTION = PARIS (-CF)

where the certainty factor (CF) is negative. Initially, both kinds of rules were generated internally in ETS in a simple frame format. Then several KS300 knowledge base files were examined to discover how rules, parameters, and control information were represented. In several days ETS was generating its first knowledge bases that could be loaded directly into KS300.

Many ad hoc methods for generating certainty factors were tried. The algorithm would be changed, ETS would generate a rule base for KS300, the rule base would be loaded, and test cases would be run in KS300 to see how well the results fit the expert's expectations. Art Nagai's patience as a database expert was invaluable. Eventually, an algorithm was used that employed the weight of the trait (assigned by the expert in ETS) and the relative strength of the rating (a rating of 5, for example, was stronger than a rating of 4). It could compute a maximum certainty factor based on the number of rules that were generated (Boose, 1986a). Although we did not have the resources to perform a formal study, KAQ knowledge bases loaded in KS300 produced reasonable results for several applications by several experts. The algorithm continued to receive minor changes until it was replaced in 1985 in NeoETS.

The use of rules generated from implications in KS300 produced interesting results. KS300 backchained on the rules to find the most general or controlling implications and asked about these first. Answering one question during a consultation could thus provide values for several traits, in some cases dramatically reducing the number of questions KS300 needed to ask. Sometimes

this could cause problems, since the expert didn't always agree with all the implications ETS produced; the inclusion of such rules was made optional.

ETS helped people develop prototypes very quickly:

- Experts found it easy to interact directly with the tool. In fact, almost all the experts who used KAQ and ETS wanted to come back and continue using it. The interface and underlying methodology were straightforward and transparent. Knowledge was entered in the expert's vocabulary. The analysis tools offered experts insights that they found interesting and surprising. The use of these tools at the beginning of a project seemed to short-circuit typical problems brought about by manual interviewing (expert insecurity, fear of exposing problem-solving information to others, and anxiety caused by not knowing what the process was all about). This general enthusiasm of experts for using KAQ and ETS was probably one of the most important factors in their success.

- People didn't have to be experts to use ETS. People used ETS to help select workstations, classify personnel, buy cars, and improve their golf swings. ETS seemed to apply to any problem with a relatively small number of potential solutions. Even if the resulting consultation system wasn't perfect, ETS helped users derive insights from analysis tool use and helped them think about their problems in a structured manner.

- Knowledge engineers who watched experts use KAQ and ETS and examined system transcripts could quickly learn vocabulary, see potential solutions, learn the appropriate level of granularity for solving a problem, view the relative importance of initial traits, test a working prototype, and ask the expert further questions about the problem in a much more directed manner than was usually possible using regular manual interviewing methods. In fact, ETS was of great help to most Associates, since they generally had no background in formal interviewing techniques (as is still true for most knowledge engineers today).

- Rapid prototypes could be built in an hour or less. It usually took an expert a half hour or so to build an initial grid (typically from 6-by-6 to 10-by-10) and examine the implications. Then it took about ten minutes to generate rules, stop ETS, start KS300, and load the knowledge base. This minor delay was dealt with later by building a reasoning engine in ETS; see below. The expert could then build and save test cases to see how well the knowledge base performed. Problems were dealt with by reloading ETS, adding to or modifying information in the grid, and then retesting. This rapid prototyping capability led to the use of ETS for feasibility analysis of many different potential expert system projects. It was relatively easy to invite an expert over to the Center for a few hours and try out several project ideas.

- It was much easier to maintain information in a grid than in rules. Grids were a more compact form of representation and were easier to comprehend than a set of rules.

- Knowledge bases produced by ETS could be modified or combined with other knowledge base information in KS300; ETS could thus produce components of larger systems.

- ETS was a good training tool. It was easy to explain basic expert system concepts by example - ETS produced rules automatically and KS300 could run consultations. Managers and others involved in the potential project could see the working prototype and quickly get an idea about knowledge-based system technology and its problem-solving potential for a particular project.

We had also improved repertory grid methodology:

- In ETS, repertory grids were *operational*. Information contained in grids could be used by an inference engine to solve problems and help make decisions. The link to KS300 provided dynamic analysis, whereas previous methods provided only static analysis.

- In ETS grids had been applied practically to knowledge-based systems; grid techniques had been successfully applied to other areas such as training and market analysis.

- Having several kinds of analysis tools together in one system also offered significant advantages. Users could systematically transform grid information from one form to another to learn about its strengths and limits and measure different aspects of its problem-solving power.

There were, however, significant limitations to the methodology and its implementation in KAQ and ETS (Boose, 1984, 1985, 1986a). Some of these difficulties have been overcome or reduced in our subsequent work.

- The method seemed to be limited to *analysis* problems - problems for which the solutions could be enumerated comfortably. *Synthesis* problems, such as configuration, scheduling, and design for which solutions are typically built up from component parts, were difficult to address using grids. It also seemed difficult to use grid methods to elicit causal,

procedural, or strategic knowledge. Often this kind of knowledge seemed necessary for controlling reasoning in complex problems.

- There was a limit to how much information could be comfortably represented in a single rating grid. One application included a 38-by-35 grid, but it was hard for the expert to use and manage, partly because some of the analysis tools took too long and partly because it was difficult to comprehend that much information at once. Some method was needed to decompose large grids into manageable, related subgrids.

- Representing solutions at different levels of abstraction led to trouble when using analysis tools and building rapid prototypes. This became clear as one of the first Associates tried to build a jet engine diagnostic aid. Parts and systems were listed as potential solutions (problem areas), and diagnostic symptoms were generated through triadic comparison. Unfortunately, solutions such as "spark plug" and "electrical system" both appeared in the grid. A spark plug was a component of the electrical system. What was really needed was a grid that represented high-level subsystems, individual subgrids that represented components of those subsystems, and so on. In this case, the "leaf" grids would contain field-replaceable components. If such a hierarchy of grids existed, traits on the top-level grid would be symptoms of problems in overall subsystems, symptoms one level down would relate to components of subsystems, and so on. This type of problem and the problem of large grids were the major motivations for building hierarchies in NeoETS, discussed below.

- Many types of problem-solving information did not fit comfortably into the types of traits used by ETS and other repertory grid tools (typically, ordinal traits with ranges from 1 to 5). In the Database Management System Advisor, for example, one important type of trait related database management system to hardware. Nagai needed several bipolar ordinal traits to represent this in ETS, such as "runs on VAX / doesn't run on VAX," "runs on IBM / doesn't run on IBM," "runs on CDC / doesn't run on CDC," and so on. It would have been much more convenient to combine all this information into one nominal trait, "computer type," whose values could be VAX, IBM, CDC, and so on. Then these actual values, or sets of these values, could appear in the grid, rather than numbers. This type of problem was a matter of convenience; it was possible but awkward to represent such information in ETS. Another problem with ordinal ETS rating scales was one of precision. In a composite parts advisor, an Associate wanted to relate critical temperatures in a manufacturing process to potential solutions. Two problems arose. There were more than five critical temperatures (where trait ranges were only from 1 to 5), so the Associate was forced to use multiple traits to represent these temperatures ("critical point A / not critical point A," "critical point B / not critical point B," and so on). More critically, the Associate needed to represent the exact temperature, not an ordinal representation of the temperature. These types of problems led to the introduction of nominal, interval, and ratio traits in NeoETS, as well as the ability to adjust the ranges of the traits.

- Consultations produced by ETS allowed users to enter only *soft* constraints. They could specify preferences about the solution set but could not indicate that a given trait should control a situation absolutely. For example, the user could specify that cost should be low on an ordinal scale (a preference of 1 on a LOW-COST (1) / HIGH-COST (5) trait) but high cost solutions might still be selected. The user might not be able to spend more than a certain amount of money, but this could not be specified. The expert needed a mechanism for adding these *hard* constraints; such a mechanism was added in Aquinas (described below). Interactions between traits couldn't be specified either, where knowing the value of one trait limited the values of another. For instance, knowing that cost should be low might imply other features of the solutions. This information couldn't be represented in an ETS grid, but, again, was later implemented in Aquinas.

- More tools were needed to help revise an initial rating grid to increase its problem-solving performance. More static analysis was needed to explore the grid's potential power, and a feedback link was needed from the inference engine so that dynamic analysis tools could directly examine performance results from real problems. Some of these problems, mentioned below, were addressed in Aquinas.

- Kelly had pointed out problems in the psychotherapeutic domain that related directly to the use of grids for problem-solving. In particular, it was hard to tell whether or not the elements and constructs sufficiently represented the necessary information. Issues of necessity and sufficiency continue to plague knowledge engineers as they seek to verify and validate knowledge-based systems, no matter what tools they use. We have begun to add testing and performance measurement features to Aquinas, but, clearly, this will remain a difficult problem.

Before the end of 1983, other improvements were made to ETS. ETS was trained to make knowledge bases for OPS5 running on our Center's VAX. This provided an inexpensive, easily accessible environment for running consultation systems produced by ETS. Further triadic elicitation and automated implication reviewing were also implemented. ETS then contained the following additional features:

- Automatic rule generation
  Operationalize information in a repertory grid
- Rules produced in KS300 format
  Allow dynamic testing of grid knowledge - expert can agree or disagree with conclusions and improve knowledge
  Significantly expand repertory grid methods

Implication rules in KS300 cut down number of questions asked
Potentially deliver knowledge bases produced by ETS on non-Lisp machine
Expand sphere of usefulness

- Manual test review
  Provide mechanical methods for improving behavior
  Measure performance
- Rules produced in OPS5 format
  Provide VAX-based delivery of knowledge base produced by ETS
  Provide delivery in multi-user time sharing environment
- Further triads
  Further improve knowledge base from "random" triads
- Automated implication review

In November 1983, Boose gave a talk to the local Puget Sound Association for Artificial Intelligence at the University of Washington. This was the first time ETS had been presented outside Boeing. He also started a paper that was presented next summer at the American Association of Artificial Intelligence conference in Austin (AAAI-84; Boose, 1984).

Early in 1984, an internal inference engine was implemented in ETS. This relieved the annoyance of stopping ETS and starting KS300 on the Lisp machine. It also allowed the addition of debugging aids such as consultation tracing and rank-order comparison. Experts would enter their expected results and ETS would compare the list with the actual results using a simple rank correlation measure. Thus, the performance of knowledge in the rating grid could be measured, and experts could see whether changes made to a grid resulted in overall performance improvement or degradation. New interesting triads could also be formed based on the worst misplaced solution in the set (Boose, 1986a).

We also implemented laddering and trained ETS to make knowledge bases for several other knowledge-based systems shells that were in use at the Center. This resulted in the following additional capabilities:

- Embedded testing using internal inference engine
  Eliminate step of making KS300 knowledge base, switching partitions, and loading
  Allow subsequent internal test reviewing
- Internal test review for rank order performance measurement and "worst misplaced solution"
  Allow problem-directed debugging
- Laddering
  Support ambiguity analysis in implication review
  Further elicit more general or more specific traits
- Rules produced for LOOPS knowledge bases
- Rules produced for S.1 and M.1 knowledge bases
- Rules produced for TI Personal Consultant knowledge bases
  Support delivery in other hardware and software environments

In the spring of 1984, Boose visited William Clancey at Stanford. They discussed ETS and Clancey's forthcoming paper on heuristic classification. Information from this discussion later formed the basis of the reasoning structure in Aquinas.

ETS was used in the first knowledge engineering course taught at the Center. A user's manual was produced for knowledge engineers and experts. ETS was installed on several other Xerox Lisp machines as Associates graduated and returned to their home groups. A non graphic version of ETS, running under ISI Interlisp, was also ported to the Center's VAX.

In September, 1984, Boose received a letter from Mildred Shaw inviting him to visit her and Brian Gaines at York University in Toronto and to present a paper at the Personal Construct Congress in Cambridge, England, in 1985. She had seen the ETS paper in the AAAI-84 proceedings and was interested in seeing ETS and talking with us. Boose and a colleague, Ray Allis, visited Toronto in October and gave several talks and demonstrations. Shaw and Gaines showed them PLANET, an Apple II-based repertory grid tool embodying most of the features they had written about. The discussions were very fruitful, and Boose, Gaines, and Shaw formed a relationship that led to, among other things, a series of knowledge acquisition workshops (discussed below).

PLANET had several useful analysis tools that Boose implemented in ETS. One of these was similarity analysis and review, where rows and columns in the grid were measured for similarity and difference. This was a powerful static analysis technique when applied to problem-solving information in ETS. For example, if two solutions were highly similar (the ratings in their columns matched significantly), then the inference engine would not be able to discriminate between them very well. The review mechanism allowed the expert to add traits in a directed manner to help solve these problems. This same process, when applied to rows, could help find redundant or highly dependent traits.

Another item contained in PLANET was a cluster analysis tool, FOCUS. A similar tool was implemented later in NeoETS to help experts decompose large grids into hierarchies.

It was common wisdom that only one expert should be used to build an initial expert system, but many Associates' problems demanded expertise from multiple sources. Sometimes knowledge about similar solutions was needed from more than one expert (multiple experts in the same domain); sometimes different areas of expertise from several experts needed to be linked together (multiple experts in different domains). One problem with combining knowledge from different experts was that information tended to be "averaged" across experts to reduce conflict, which results in the loss of special case-information. How could knowledge be represented and used from multiple sources?

In the fall of 1984, we combined expertise from multiple experts using ETS (Boose, 1986b). Expertise was first elicited individually from experts. Then ETS produced S.1 knowledge bases from the individual experts' grids, and the knowledge bases were manually pasted together and loaded in S.1. Each rule was produced with an additional screening clause that recorded which expert had contributed the knowledge. The resulting consultation system first asked the user for the set of experts to be considered; the user could weight experts by applying certainty factors to the response. Then S.1 backchained as usual to produce consensus consultation results. An S.1 mechanism was added to measure each expert's distance from the consensus, and the "most dissenting" expert's opinion was then displayed side by side with the consensus. This allowed the user to see a full range of opinion, not just the consensus average. Experts could also use their own sets of solutions and traits; S.1 would ask about each necessary trait during backchaining. To the extent that experts used similar vocabulary and problem-solving models, the user could more easily compare their individual results.

This method combined a delphi-like approach to gathering information independently with the rapid elicitation capabilities of repertory grids. Since each expert's knowledge was preserved, no special-case information was lost. Information could be gathered rapidly from a large group of experts. Later, we implemented this method directly in NeoETS (discussed below), along with analysis tools for comparing grids from different experts.This capability led to subsequent applications of NeoETS and Aquinas for group consensus and group decision making.

By the end of 1984, then, ETS also contained these features:

- Similarity analysis and review (from PLANET)
  Measure static problem-solving potential of a grid
  Provide specific help for increasing discriminating power of grid
  Provide method for adding new solutions and traits in a directed manner
  Measure trait subsumption and independence
- Multiple expertise knowledge bases built in S.1 and OPS5
  2200-rule AI tool advisor produced for OPS5/VMS
  Combine different sources on same or different topics

In December 1984, Gaines and Shaw visited our Center and gave a colloquium on their views of knowledge engineering and personal construct methods. Plans were made for a knowledge engineering workshop at York University the following May.

By the end of the year, ETS had been used to produce hundreds of prototype knowledge-based systems and had begun to receive wide publicity within the AI community. Some of these applications included an AI library advisor, aircraft fault identifier, airplane design flutter analyzer, automated numerical control cutter consultant, flight controls diagnostic aid, ATLAS structural analysis advisor, bond durability consultant, business computing needs advisor, business graphics package consultant, composite materials advisors, documentation update consultant, energy control system model evaluator, failure modes and effects analyzer, finish advisor for materials, flight mode manager, materials technology advisor, mental health diagnostic aid, microcomputer needs analysis, molded rubber seal advisor, office automation system advisor, organization development intervention consultant, PAN AIR software aids, portfolio management aid, programming bug finder, programming language advisor, propulsion system advisor, rivet selector, robotic system application advisor, software management consultant, software release advisor, technical sales and services consultant, velocity analysis advisor, wine advisor, and word processing system consultant.

5. NEOETS: HIERARCHICAL REPERTORY GRIDS AND MULTIPLE DATA TYPES

In January 1985, Boose taught a knowledge engineering course that was attended by a University of Washington psychology graduate student, Jeff Bradshaw. Already familiar with repertory grids, Bradshaw became interested in ETS and started working at the Center in the spring. Bradshaw contributed key ideas to NeoETS in the areas of knowledge representation and reasoning.

By the fall of 1985, NeoETS had been born in response to many of the limitations of ETS, especially those related to large grids, problem decomposition, levels of abstraction, and multiple data types. In NeoETS, we attempted to augment repertory grids with new representations (hierarchies and multiple data types) and a fuller set of integrated analysis and reasoning tools. The most significant change was a de-emphasis of rule generation and an effort to deliver knowledge to end users within Aquinas.

The internal structure of ETS was removed and replaced several times to produce a system that could handle multiple hierarchical repertory grids, multiple data types, new analysis tools, and an internal reasoning engine for these new structures. We began

referring to NeoETS as a *knowledge acquisition workbench*. The additional complexity created new challenges for effective elicitation, analysis, and testing of knowledge. Features of NeoETS are documented in (Bradshaw and Boose, 1986). They include:

- New data structures
  - Four hierarchies - solutions, traits, cases, and experts
    - *Allow for complexity, multiple levels of abstraction*
    - *Handle multiple experts within one NeoETS knowledge structure*
    - *Partition knowledge around specific subproblems (cases)*
    - *Handle hierarchical relationships in any data type*
  - Four trait scale types - nominal, ordinal, interval, ratio
    - *Handle trade-offs between precision, convenience, conciseness, cost*
- Reasoning engine based on the Analytic Hierarchy Process (Saaty, 1980)
  - *Serve as first attempt at reasoning with hierarchical structures*
- Additional bridges produced for other shells (KEE; an internal C-based shell)

In May, a knowledge engineering seminar was held at York University, and later a new article about ETS provided more detail on the methods and their implementations. Work was also started on a book that was published early in 1986 (Boose, 1986a).

In the fall of 1985 we started a project with Seattle University's software engineering master's program that was to deliver a C-based version of ETS in the spring of 1986. This project was successfully completed, and we have continued with follow-on projects each year to enhance the utility of this tool. Currently, many of the useful tools in the Lisp-based version of Aquinas have migrated to the C-based version.

6. AQUINAS: A WORKBENCH FOR KNOWLEDGE BASE LIFE CYCLE SUPPORTAQUINAS DEVELOPMENT HISTORY

Further developments of our workbench were aimed at handling knowledge base life cycle problems, the user interface, delivery of knowledge bases (and of the workbench itself), and integration with other tools and systems.

1986

Early in 1986, David Shema and Cathy Kitto joined our project. Shema worked initially at improving the user interface and Kitto began work on a dialog manager to help the user handle the growing complexity of the workbench (Kitto and Boose, 1987). Shema eventually started work on validation and verification methods, implementing a case storage and batch-replay mechanism that could be used to measure knowledge base performance and even automatically improve information in rating grids (Shema and Boose, 1988).

The initial dialog manager was an expert system within Aquinas that used knowledge acquisition heuristics (about knowledge engineering, knowledge acquisition cycles in Aquinas, uses and limits of Aquinas tools, and idiosyncratic knowledge) and information about the current session and knowledge base complexity and completeness. Later, the dialog manager was expanded to help the user select an overall category for an application problem (Kitto and Boose, 1988). This would help the user decide 1) if Aquinas was appropriate for the problem and 2) what knowledge-based system tool might be useful. The dialog manager rank-ordered appropriate potential tools to use in Aquinas at each interaction cycle and suggested several to the expert. The dialog manager was difficult to maintain, as other team members constantly changed tools and the information that led to tool choices became outdated.

For reasons that should be apparent, we were unhappy with the name "NeoETS" and renamed the system Aquinas. From 1986 to the present, we added numerous additional tools and techniques (summarized in Boose, Shema, and Bradshaw, 1988); a few of the more interesting developments are mentioned here.

Bradshaw, dissatisfied with the use of the Analytic Hierarchy Process for reasoning (Saaty, 1980), implemented an extended version of Clancey's heuristic classification model (Clancey, 1986) that included a maximum-entropy-based method for propagating uncertainty (Boose and Bradshaw, 1987). An important aspect of this method included techniques for performing inductive inference in sparse hierarchies of grid information. Bradshaw also implemented such useful features as a cluster analysis based on FOCUS and the ability to represent discrete probability distributions in the cells of a rating grid.

Tools were developed, enhanced, or integrated to help experts structure information in hierarchies by noticing trait relationships and patterns in ratings (cluster analysis, laddering, trait value examination, Hinkle's pattern analysis, automatic or controlled inheritance, graphic cut and paste). Other tools were developed to analyze similarities and differences across groups of experts.

Table completion and boundary analysis were developed to help check grids statically for consistency and completeness (Boose and Bradshaw, 1987).

In November 1986, the first AAAI-sponsored knowledge acquisition workshop was held in Banff, Canada, co-chaired by Boose and Gaines. This was the first in a series of annual workshops in Banff and Europe that provided a unique forum for leading knowledge acquisition theoreticians and practitioners (Boose and Gaines, 1989).

1987

In early 1987, Bradshaw implemented a form of ID3 (Quinlan, 1988) in the inference engine. It performed a cost-benefit analysis on repertory grids that allowed the inference engine to order consultation questions dynamically in a best-first manner.

Multiple expert capabilities mentioned above were added to the knowledge base structure and the inference engine. Users could select subsets of solutions and weight experts. Consultation results showed the consensus and the dissenting opinion. Tools for analyzing the differences and similarities between experts were added.

Boose documented the large number of applications and problems that had put KAQ, ETS, NeoETS, and Aquinas through their paces and reviewed uses of Aquinas to elicit procedural and strategic knowledge (Boose, 1988a). He implemented an on line help mechanism and documented the growing number of knowledge acquisition tools and techniques exposed at the knowledge acquisition workshops (Boose, 1988b).

Shema developed techniques for debugging knowledge bases and measuring knowledge base performance. Manual, directed, and automated methods were implemented in an effort to provide validation and verification tools for life-cycle support (Shema and Boose, 1988). Later trait sensitivity and solution stability analysis mechanisms were added.

In 1987, Bradshaw started a new project combining ideas from decision analysis and knowledge-based systems (Howard and Matheson, 1984; Holtzman, 1989). This led to interesting theoretical work on integrating repertory grids with knowledge structures used in decision analysis, such as influence diagrams, a kind of Bayesian network (Bradshaw and Boose, 1988; Bradshaw, Boose, Covington, and Russo, 1989a,b). Kitto also started her own project using KNACK, a knowledge acquisition tool originally developed at Carnegie Mellon (Kitto, 1988; Klinker et al., 1989).

1988

In 1988, Shema implemented a constraint mechanism that allowed experts and users to represent trait interactions and hard constraints (Boose, Shema, and Bradshaw, 1988). He added a consultation spreadsheet mechanism that allowed users to easily reason about hypothetical situations by temporarily modifying grid ratings and weights and immediately seeing graphic consultation results. Another feature added by Shema was a link to a graphic database so that images could be shown and used during consultations. He has been instrumental in improving the user interface with graphic display and input tools and screen organization devices.

More improvements were made to the internal reasoning engine. The algorithm was improved, and hooks were added so that experts and users could easily tailor certain steps in the inference process based on particular applications.

Initial work was performed on capturing explanations from experts during knowledge elicitation and on keeping a history of knowledge base changes. We believe that explanations will be useful as an aid for knowledge base maintenance and for allowing end users to review the expert's justifications.

Aquinas has long been used to "try out" different values of trait ratings, weights, and preferences to see how the recommendations are affected. Spreadsheet consultations were added in the Fall of 1988 which allow the experts or users to temporarily change values and observe how the consultation recommendations would vary.

Work has been done to integrate Aquinas into other environments and tools. In the summer of 1988, Envos, a Xerox spin-off company, announced that the entire Xerox-based programming environment was available on Sun workstations. Now, the full version of Aquinas is available to anyone in Boeing with a Sun workstation. The C-based version of Aquinas continues to be moved to different workstations. A recent version, MacQuinas, which runs on Macintosh computers, is being integrated with MacXotl (the Mac II version of Axotl, a knowledge-based decision analysis workbench implemented in SmallTalk-80) within an environment named "folie a deux" (Bradshaw, Boose, Covington, and Russo, 1989a,b). This will allow us to combine repertory grid and decision analysis techniques in one framework.

Many of the limitations of repertory grid methods in general and our implementations in particular have, at least in part, been overcome. However, several major problems are yet to be solved. These are discussed in the next section.

7. THE FUTURE: EMBEDDED KNOWLEDGE, SYNTHESIS PROBLEMS, AND LIFE CYCLE SUPPORT

Aquinas and Other Knowledge Acquisition Tools

Most knowledge acquisition tools derive their power from application domain knowledge. Figure 1 shows representative knowledge acquisition tools plotted on two axes: domain independence and problem type (discussed in detail in Boose, 1988b). Tools such as FIS and STUDENT are built for single-application systems. They are successful because their developers could embed very specific knowledge about the domain application, but they are difficult to generalize to new applications. Tools such as MOLE, MDIS, TKAW, and SALT derive power from knowing about patterns of knowledge types and problem-solving methods in certain classes of domains. Although they can be generalized within their target subdomains, developers find it difficult to generalize them to relatively unconnected domains.

On the other hand, tools like KSS0 and Aquinas have no embedded domain knowledge. They are very general but lack domain power.

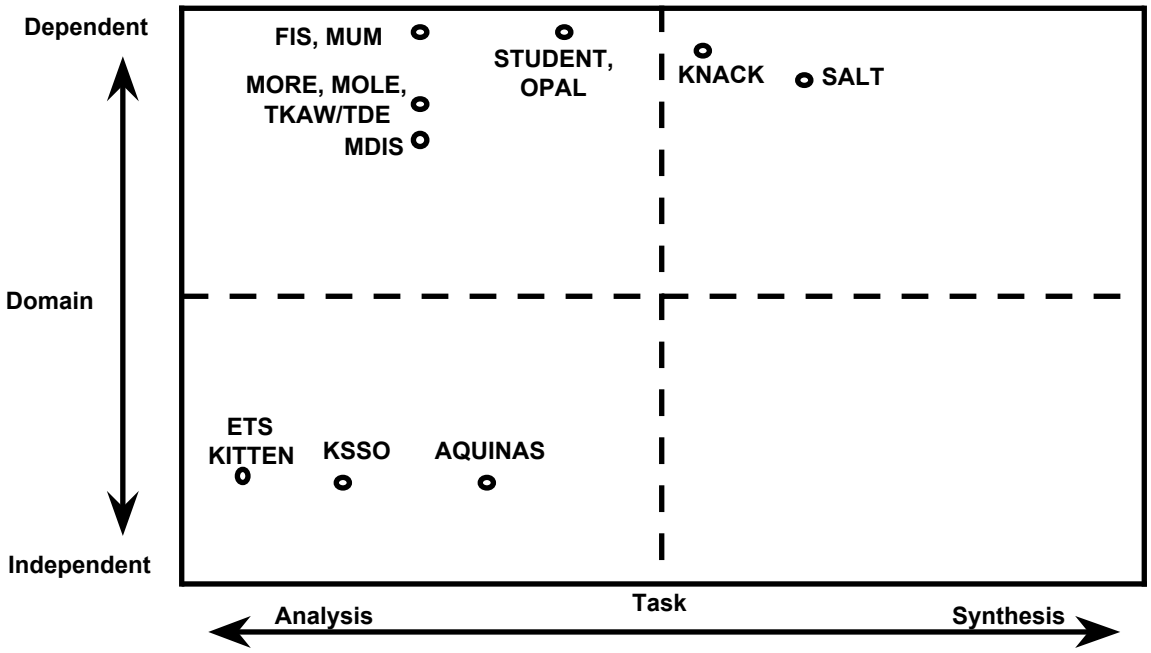

Figure 1. Knowledge acquisition tools plotted on two axes: *domain independence* and *problem type*.

One approach to improving Aquinas would be to develop domain knowledge templates that could help an expert start building a knowledge base (Bradshaw, Boose, Covington, and Russo, 1989b). One obvious area would be diagnostic problems where much is known about how to link symptoms, tests, costs of tests, and hypotheses. Loading such domain knowledge in Aquinas would significantly expand its problem-solving power.

Synthesis Problems

As can be seen in Figure 1, few knowledge acquisition tools have been successful at handling synthesis problems, and those that do are domain specific (Boose, 1988b).

We are trying to expand Aquinas to be able to handle simple synthesis problems such as configuration and process monitoring. For example, we are allowing users to put arbitrary computations in the cells of repertory grids in AQUINAS (spreadsheet-like computations, database access calls, functions that sample sensors) or to call specialized tools for alternatives generation and constraint satisfaction (Bradshaw, Boose, Covington, and Russo, 1989a; Shema, Covington, Boose, and Bradshaw, 1989).

Eventually, we hope to blend techniques from other tools (such as MDIS, Antonelli, 1983, and SALT, Marcus et al., 1985; Marcus, 1987) to help elicit knowledge for aspects of design problems such as:

- Acquisition of causal models and design constraints.
- Selecting alternative design concepts based on competing criteria such as reliability, maintainability, cost, and manufacturability.

- o Acquiring human expertise in validation and verification of designs.
- o Acquiring knowledge for design evaluation and change recommendation.
- o Acquiring knowledge for document generation and evaluation.
- o Capturing historical data in a "corporate memory" database to help solve future design problems.

We also intend to continue work on eliciting strategic and procedure knowledge to help control complex problems. While the dialog manager was successful in providing help to people learning Aquinas (Kitto and Boose, 1987), the rule-based format of its knowledge base was difficult to maintain. We are currently evaluating the use of repertory grids (Boose, 1988) and extended AND-OR graph representations (Bradshaw, Boose, Covington, and Russo, 1989a,b) to represent strategic and procedural knowledge.

Life Cycle Support

Figure 2 illustrates our intentions to increase Aquinas' knowledge acquisition power by embedding domain-specific knowledge and by augmenting Aquinas structures to handle synthesis problems.

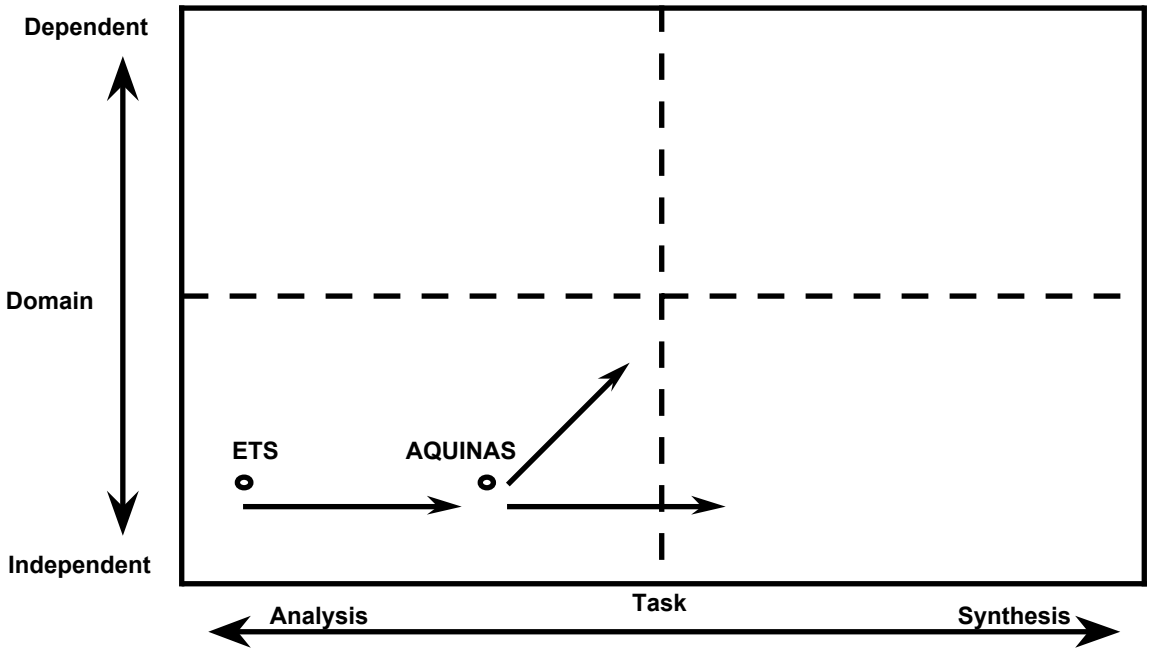

Figure 2. In the future we intend to expand Aquinas's knowledge acquisition power by embedding domain-specific knowledge and by augmenting Aquinas structures to handle synthesis problems.

In addition, we will attempt to support more of the knowledge-based system life cycle. We will concentrate on the areas of validation, verification, and maintenance of knowledge (Baum, Shema, and Boose, 1989).

In the near future, we will continue to build on Shema's work to measure and improve knowledge base performance, particularly in managing long-term maintenance of knowledge bases built by multiple experts. We will explore tools that tailor reports for maintainers after changes are made and perform sensitivity analyses to identify critical areas of knowledge. We may also implement TEIRESIAS-like features to model existing knowledge and identify inconsistencies when changes are made.

We will also explore tools for text analysis (Gaines, 1987) and improve our earlier work on explanation capture and playback mechanisms.

In the long term, we hope to provide full delivery facilities, including knowledge base partitioning and protection,and further facilities for eliciting and analyzing knowledge from multiple experts (elicit knowledge in parallel (on line) from several experts, analyze subsumption and overlap, support on line structured negotiation, perform intelligent knowledge base merging, discover the most important aspects of individual knowledge (special cases), and elicit and merge knowledge dynamically).

We are continuing our relationship with Gaines and Shaw, who are now at the Knowledge Science Institute at the University of Calgary, Alberta, Canada. Gaines and Boose have edited special knowledge acquisition issues of the *International Journal of Man-Machine Studies*, and are co-editing the Knowledge-Based System book series series for Academic Press, starting a new journal, *Knowledge Acquisition: An International Journal*, from Academic Press, and continuing to co-chair annual knowledge acquisition workshops in North America, Europe, and Japan.

8. CONCLUSION

Two major forces brought about the development of our knowledge acquisition tools : technology push and application pull. Ideas from many areas were gradually integrated to meet the growing demands of knowledge-based systems problems at The Boeing Company. This paper briefly traced the history of this development, and the sources of many of our ideas, described features and the reasons that they were added, and illustrated typical applications at each stage of evolution.

Building useful knowledge acquisition tools continues to provide exciting challenges to make knowledge-based systems and knowledge-based decision aids ever more practical. ETS and Aquinas have contributed significantly to the use of knowledge-based systems within The Boeing Company.

ACKNOWLEDGMENTS

Thanks to Miroslav Benda, Kathleen Bradshaw, Stan Covington, Brian Gaines, Sandy Marcus, Art Nagai, Peter Russo, Doug Schuler, Mildred Shaw, Suzzanne Shema, Lisle Tinglof-Boose, and Bruce Wilson for their contributions and support. KAQ, ETS, NeoETS, and AQUINAS were developed at the Boeing Advanced Technology Center in Boeing Computer Services, Seattle, Washington.

REFERENCES

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, CO.

Baum, L. S., Shema, D. B., and Boose, J. H. (1989). Acquiring and Verifying Control Knowledge for a Blackboard System, *Proceedings of the 4th AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, October, in preparation.

Boose, J. H. (1984). Personal Construct Theory and the Transfer of Human Expertise, *Proceedings of the National Conference on Artificial Intelligence (AAAI-84)*, Austin, Texas.

Boose, J. H. (1985). A Knowledge Acquisition Program for Expert Systems Based On Personal Construct Psychology, *International Journal of Man-Machine Studies*, Vol. 23.

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

Boose, J. H. (1986b). Rapid Acquisition and Combination of Knowledge from Multiple Experts in the Same Domain, *Future Computing Systems Journal*, Vol. 1, No. 2.

Boose, J. H. (1988a). Uses of Repertory Grid-Centred Knowledge Acquisition Tools for Knowledge-Based Systems, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, Vol. 29, No. 3, pp. 287-310.

Boose, J. H. (1988b). A Survey of Knowledge Acquisition Techniques and Tools, *Proceedings of the 3rd Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November 1988, University of Calgary SRDG Publications, Department of Computer Science; also in *Knowledge Acquisition: An International Journal*, in press.

Boose, J. H., and Bradshaw, J. M. (1987). Expertise Transfer and Complex Problems: Using Aquinas as a Knowledge Acquisition Workbench for Expert Systems, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 1, *International Journal of Man-Machine Studies*, Vol. 26, No.1.

Boose, J. H., and Gaines, B. R. (1989). A Summary of the First and Second Knowledge Acquisition for Knowledge-Based Systems Workshops, *AI Magazine*, in press.

Boose, J. H. Shema, D. B., and Bradshaw, J. M. (1988). Recent Progress on Aquinas: A Knowledge Acquisition Workbench, *Proceedings of the 3rd Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November, 1988, University of Calgary SRDG Publications, Department of Computer Science.

Bradshaw, J. M., and Boose, J. H. (1986). NeoETS: Interactive Expertise Transfer for Knowledge-Based Systems, *Proceedings of the Symposium on Human-Computer Interaction and Cognitive Engineering* at the 1986 International Conference on Systems, Man, and Cybernetics, Atlanta, GA, October.

Bradshaw, J. M., and Boose, J. H. (1988). Decision Analytic Techniques for Knowledge Acquisition: Combining Information and Preference Models Using Aquinas, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Bradshaw, J. M., Boose, J. H., Covington, S. P. and Russo, P. J. (1989a). How To Do with Grids What People Say You Can't, *Proceedings of the 3rd Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November 1988, University of Calgary SRDG Publications, Department of Computer Science.

Bradshaw, J. M., Boose, J. H., Covington, S. P. and Russo, P. J. (1989b). Folie a Deux: Integrating Aquinas, a Personal Construct-Based Knowledge Acquisition Workbench, with Axotl, a Knowledge-Based Decision Analysis System, submitted to the 3rd European Knowledge Acquisition Workshop, Paris, July, 1989.

Clancey, W. J. (1986). Heuristic Classification, in J. Kowalik (ed.), Knowledge-Based Problem Solving, New York: Prentice-Hall.

Davis, R., and Lenat, D. B. (1982). Knowledge-Based Systems in Artificial Intelligence, New York:McGraw-Hill.

Gaines, B. R., and Shaw, M. L. G. (1981). New Directions in the Analysis and Interactive Elicitation of Personal Construct Systems, in M. L. G. Shaw (ed.), Recent Advances in Personal Construct Technology, New York: Academic Press.

Gaines, B. R. (1987). Advanced Expert System Support Environments, in Boose, J. H. and Gaines, B. R. (eds.), *Proceedings of the Second Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, October.

Hinkle, D. N. (1965). The Change of Personal Constructs from the Viewpoint of a Theory of Implications. PhD Dissertation, Ohio State University.

Holtzman, S. (1989). Intelligent Decision Systems, Reading, MA: Addison-Wesley.

Howard, R. A. and Matheson, J. E. (1984). Readings on the Principles and Applications of Decision Analysis, Menlo Park, CA: Strategic Decisions Group.

Keen, T. R., and Bell, R. C. (1981). One Thing Leads to Another: A New Approach to Elicitation in the Repertory Grid Technique, in M. L. G. Shaw (ed.), Recent Advances in Personal Construct Technology, New York: Academic Press.

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

Kitto, C., and Boose, J. H. (1987). Heuristics for Expertise Transfer: The Automatic Management of Complex Knowledge Acquisition Dialogs, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 2, *International Journal of Man-Machine Studies*, Vol. 26, No. 2.

Kitto, C. M., and Boose, J. H. (1988). Selecting Knowledge Acquisition Tools and Strategies Based on Application Characteristics, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Klinker, G., Genetet, S. and McDermott, J. (1989). Knowledge Acquisition for Evaluation Systems, *International Journal of Man-Machine Studies*, in press.

Marcus, S., McDermott, J., and Wang, T. (1985). Knowledge Acquisition for Constructive Systems, *Proceedings of the Ninth Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, pp. 637-639.

Marcus, S. (1987). Taking Backtracking with a Grain of SALT, special issue on the 1st Knowledge Acquisition for Knowledge-Based Systems Workshop, 1986, Part 3, *International Journal of Man-Machine Studies*, Vol. 26, No. 4, pp. 383-398; also in Boose, J. H., and Gaines, B. R. (eds.), Knowledge-Based Systems, Vol. 2: Knowledge Acquisition Tools for Expert Systems, New York: Academic Press, 1988, pp. 211-226.

Quinlan, J. R. (1988). Simplifying Decision Trees, in Gaines, B. R. and Boose, J. H. (eds.), Knowledge-Based Systems, Vol. 1: Knowledge Acquisition for Knowledge-Based Systems, London: Academic Press.

Saaty, T. L. (1980). The Analytic Hierarchy Process, New York: McGraw-Hill.

Shaw, M. L. G. (1980). On Becoming a Personal Scientist, London: Academic Press.

Shaw, M. L. G. (ed.), (1981). Recent Advances in Personal Construct Technology, New York: Academic Press.

Shema, D. B., and Boose, J. H. (1988). Refining Problem-Solving Knowledge in Repertory Grids Using a Consultation Mechanism, special issue on the 2nd Knowledge Acquisition for Knowledge-Based Systems Workshop, 1987, *International Journal of Man-Machine Studies*, in press.

Shema, D. B., Bradshaw, J. M., Covington, S. P., and Boose, J. H. (1989). Design Knowledge Capture and Alternative Generation Using Possibility Tables in CANARD, *Proceedings of the 4th AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, October, in preparation.

Stewart, V., and Stewart, A. (1981). Business Applications of Repertory Grids, London: McGraw-Hill.