Automating Software Design for Knowledge-Based Systems

Brian R Gaines and Mildred L G Shaw
Knowledge Science Institute
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Abstract

The development of knowledge-based systems is being treated as a new problem with its own techniques, tools and research communities. Knowledge acquisition and knowledge engineering methodologies are being developed independently of existing systems analysis and software engineering methodologies. This paper gives an overview of developments in knowledge acquisition for knowledge-based systems and compares and contrasts them with standard software engineering approaches. It is suggested that much could be gained through increased interaction between the research and user communities involved, and through the increased integration of knowledge engineering and software engineering methodologies and tools.

1 Introduction

Software engineering for knowledge-based systems development has developed outside the framework of most existing software engineering methodologies. The concepts of "artificial intelligence" and "expert systems" have been taken to have sufficient novelty to justify radically different approaches under the heading of "knowledge acquisition" and "knowledge engineering." There have now been a number of major conferences on knowledge acquisition for knowledge-based systems (Boose & Gaines 1986, 1987, 1988, Addis, Boose & Gaines 1987, Boose, Gaines & Linster 1988, Boose, Gaines & Ganascia 1989) and, while some significant differences are apparent, many of the problems of knowledge engineering are ones already addressed in software engineering. Some of the "knowledge acquisition" tools that have been developed are also applicable to the development of decision-support and database systems.

This paper gives an overview of developments and problems in software engineering for knowledge-based systems with the objective of promoting increasing interchange between the software engineering and knowledge acquisition communities.

2 The Concept of Knowledge Engineering

The original stereotype for knowledge acquisition was that the knowledge engineer elicited knowledge from an expert, encoded it in a computational form in a knowledge base, and made it available to clients for consultation through an expert system shell as shown in Figure 1. The knowledge engineer's job was "to act as a go-between to help an expert build a system...helping the expert to structure the domain knowledge, to identify and formalize domain concepts" (Hayes-Roth, Waterman and Lenat 1983). As experience in the development of knowledge-based systems has grown, problems have arisen in the role of the knowledge engineer as a go-between:

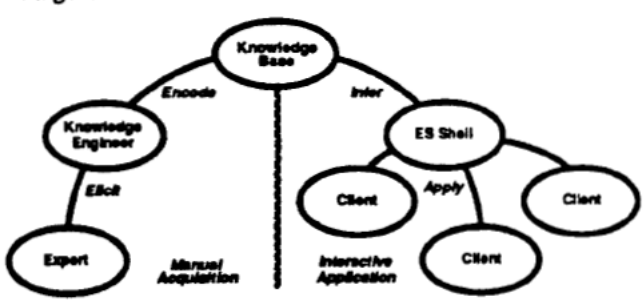

Fig.1 Original concept of knowledge acquisition

- The decline in costs of both hardware and software support for expert systems has brought the technology into a mass-market situation far more rapidly than originally envisioned.
- This has lead to a growth in demand for expert systems that is proceeding far more rapidly than the growth in supply of trained and experienced knowledge engineers.
- The declining costs of expert system technology are also making the expense of human labour in tailoring the technology for particular applications appear to be the dominating constraint and an excessive cost.
- A move towards a labor-intensive activity is contrary to all trends in industry.
- In particular it is contrary to the trend towards automatic programming techniques in the computing industry.
- The role of the knowledge engineer as an intermediary between the expert and the technology is being questioned not only on cost grounds but also in relation to its effectiveness—knowledge may be lost through the intermediary and the expert's lack of knowledge of the technology may be less of a detriment than the knowledge engineer's lack of domain knowledge.

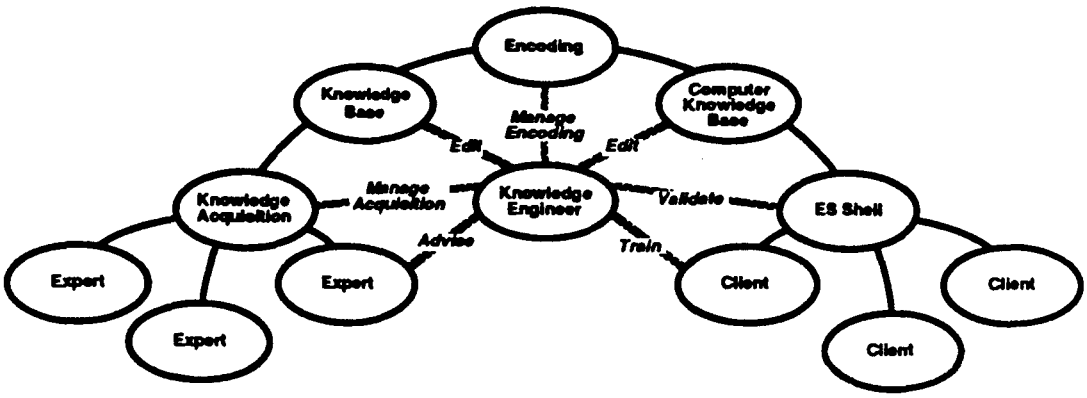

Fig.2 Extended model for knowledge acquisition

As shown in Figure 2, the knowledge engineers' roles in this extended model include:

- Advising the expert on the process of interactive knowledge elicitation.
- Managing the interactive knowledge acquisition tools, setting them up appropriately.
- Editing the unencoded knowledge base in collaboration with the expert.
- Managing the knowledge encoding tools, setting them up appropriately.
- Editing the encoded knowledge base in collaboration with the expert.
- Validating the application of the knowledge base in collaboration with the expert.
- Training the clients in the effective use of the knowledge base in collaboration with the expert by developing operational and training procedures.

This use of interactive elicitation can be combined with manual elicitation, and with the use of the interactive tools by the knowledge engineer rather than, or in addition to, the expert. Thus, the role also includes:

- Directly eliciting knowledge from the expert.
- Using the interaction elicitation tools to enter knowledge into the knowledge base.

3 Structure of Knowledge-Based Systems

One source of problems in software engineering for knowledge-based systems is the wide range of technologies involved, from Lisp machines, through production systems, expert systems shells and knowledge acquisition tools. Knowledge-based systems are high in the *virtual machine hierarchy* which has developed in computing, and there are many layers of technology between an expert system and the computer on which it is running. In theory, system developers should not need to know about the lower levels of the hierarchy—machine architectures, operating systems and implementation languages are remote from knowledge processing. However, in practice, these lower levels are the foundations on which systems are built and any defects in them can undermine the functionality of the upper levels.

Figure 3 shows significant levels of the virtual machine hierarchy for knowledge-based systems:

- *Machine Architecture*: at the lowest level of the hierarchy is the machine on which the knowledge-based system runs. This is really irrelevant, but use of Lisp in the early days of expert systems has focused attention on Lisp machines. Nowadays any good workstation is adequate.
- *Operating System*: at the second level of the hierarchy is the operating system within which the implementation runs. This needs to provide good interfaces to other programs, large databases and communications.
- *Implementation Language*: at the third level of the hierarchy is the implementation language which actually interfaces to the computer. This tended to be Lisp in the early days of expert systems but as speed and space efficiency have become significant, and knowledge representation has become better understood, other languages that support dynamic data structures such as C and Pascal have become widely used. In particular the object-oriented versions of these languages are well-suited to knowledge-based system implementation.
- *Knowledge-Based System Shell Development Language*: at the fourth level of the hierarchy is the language in which the knowledge-based system shell is written, generally a special-purpose environment for coping with knowledge representation and inference.
- *Knowledge-Based System Shell*: at the fifth level of the hierarchy is the expert system shell as a run-time environment that elicits problem-specific information from the user, provides advice based on its knowledge base, and explains that advice in as much detail as required.
- *Knowledge-Based System Support Environment*: at the sixth level of the hierarchy is the equivalent of the Application Programing Support Environment in conventional systems. The Knowledge-Based System Support Environment provides editing, browsing and debugging tools for the knowledge base. This level encompasses the major differences between the high-quality knowledge-based system shells that support major system development through extensive support tools, and the simpler shells that do not.

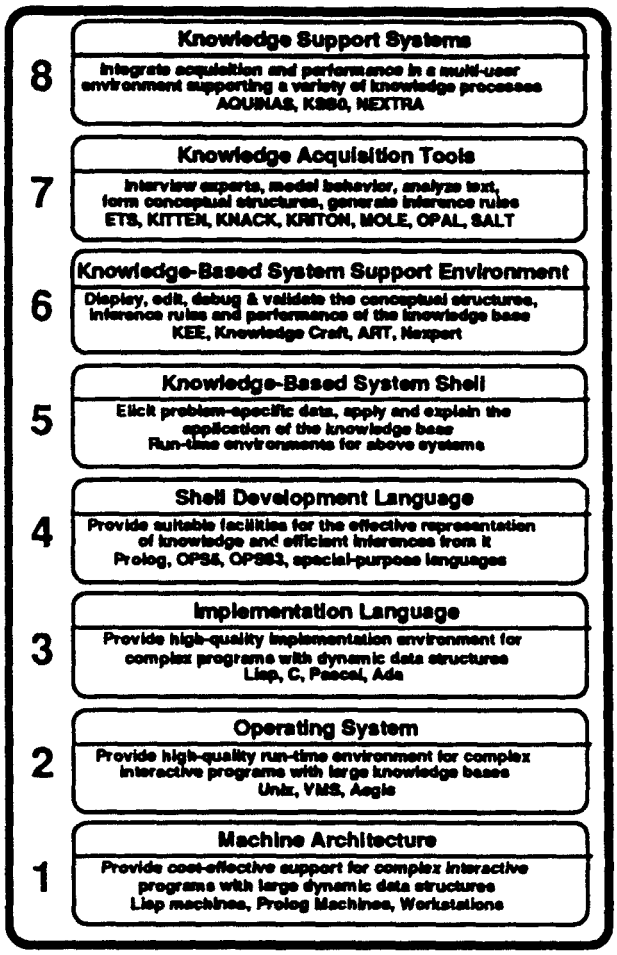

Fig.3 Virtual machine hierarchy for knowledge-based systems

- *Knowledge Acquisition Tools*: at the seventh level of the hierarchy are the tools for automating the systems analysis for expert systems, through automatic interview procedures, modeling actual expert behavior, and analysis of knowledge in textual form.
- *Knowledge Support Systems*: at the top of the hierarchy represent the objectives of recent research in knowledge acquisition that focuses on the integration of acquisition and performance systems to provide an environment supporting a wide variety of knowledge processes.

4 Knowledge Acquisition

The knowledge support and acquisition tools at the top levels of Figure 3 are the most innovative contribution of knowledge-based systems developments to software engineering. Knowledge acquisition and transfer processes in human society are complex and poorly understood (Gaines 1986). Many disciplines are involved in their study but none provides a comprehensive framework let alone overall theory. Neurology, psychology, linguistics, education, sociology, anthropology, philosophy and systems theory all have significant contributions to make.

Integrating them, disambiguating jargon, and combining different objectives and perspectives are major intellectual problems. The task is attracting increasing effort but this is unlikely to have significant short-term impact on knowledge-based system development. However, the underlying conceptual framework being used in most tools can be simply explained and contrasted with classical system development methodologies.

The center column of Figure 4 shows a system for which a decision or control system is to be designed together with sources of knowledge about the system.

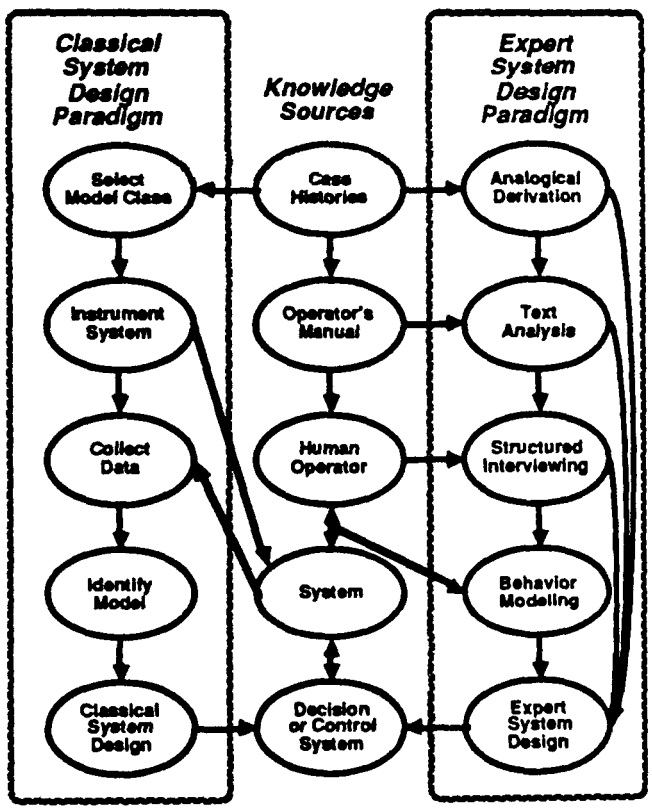

Fig.4 Classical and expert system design paradigms

The left hand column shows the use of these sources in the classical system design paradigm:

- Knowledge of past case histories is used to select a class of system models
- The information required to discriminate within this class determines how the system should be instrumented for data acquisition
- Data is collected from the system through the instrumentation
- A model is identified from the model class which best fits the data
- This model is used to design a decision or control system for optimal performance

The expert systems paradigm may be seen to be particularly applicable when it is not possible to model the system but there is an alternative source of data available because human operators are able to performance the decision or control task. The right hand column shows the use of knowledge sources in the expert system design paradigm:

- Structured interviewing may be used to acquire knowledge directly from the operators
- Behavior modeling may be used to identify the operators' strategies even if they are unaware of them or give incorrect ones in interviews
- Text analysis may be used with instructional material such as the operators' manuals
- Reasoning by analogy may be used based on the case histories without the data collection of the classical system design paradigm

Note that the classical and expert system design paradigms, and the various acquisition techniques for expert systems, need not be regarded as competitive. They may all be used in the system design, and it may be that a collection of weak methods is necessary because there is no overall strong method that can be applied.

5 Knowledge Acquisition Tools

A number of tools have been developed for the rapid prototyping of expert systems automating some or all of the methodologies on the right of Figure 4. The objective of systems such as PLANET (Shaw & Gaines 1983), ETS (Boose 1984), MOLE (Eshelman, Ehret, McDermott & Tan 1986), SALT (Marcus 1986), KITTEN (Shaw & Gaines 1986), KNACK (Klinker Bentolila, Genetet, Grimes & McDermott 1986), KRITON (Diederich, Ruhmann & May 1986), OPAL (Musen, Fagan, Combs & Shortliffe 1986), AQUINAS (Boose & Bradshaw 1987) and KSS0 (Gaines 1987a) is to expedite the process of knowledge acquisition and transfer to knowledge-based systems.

Figure 5 shows the architecture of one system that integrates a wide range of knowledge acquisition methodologies. At the heart of KSS0 is an object-oriented knowledge base in which knowledge is formally represented as a multiple-inheritance digraph of classes, objects and properties. Such a structure generalizes the entity-attribute grids used in several early knowledge acquisition systems (Shaw & Gaines 1983, Boose 1984) and has proved both general and powerful in a variety of applications (Boose & Bradshaw 1986, Gaines 1987a). The main tools in KSS0 are:

- **LINK** provides a hypertext/image notebook allowing knowledge to be entered in free form. In particular, this allows more formal structures to be annotated with explanations of vocabulary and lines of reasoning.
- **TEXAN** accepts text input directly and from the notebook, analyzes it by word frequency and association and produces a conceptual clustering of the most frequent highly-linked words. This is presented to the experts for click-and-drag graphic editing, and used to suggest sub-domains and associated entities and attributes.
- **ELICIT** accepts specifications of entities within a sub-domain and provides an interactive graphical elicitation environment within which the experts can distinguish entities to derive their attributes. The resultant class is continuously analyzed to provide feedback prompting the expert to enter further entities and attributes.

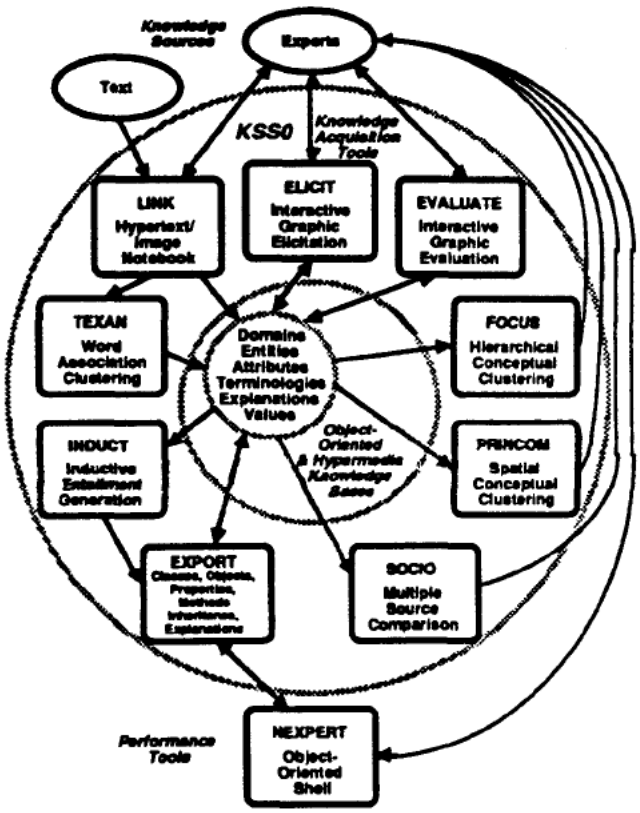

Fig.5 KSS0—an integrated knowledge acquisition system

- **EVALUATE** accepts specifications of decision-making situations in terms of specifications of relevant classes, generally of problems and potential solutions. It provides an interactive graphical elicitation environment within which evaluative attributes can be added, either by the expert,s or by the knowledge engineers from protocols of their behavior.
- **FOCUS** hierarchically clusters entities and attributes within a sub-domain prompting the experts to add higher-level entities structuring the domain.
- **PRINCOM** spatially clusters entities and attributes within a sub-domain prompting the experts to add higher-level entities structuring the domain.
- **INDUCT** induces logical entailments enabling the attributes of an entity, or the evaluations of a decision-making situation in a domain, to be derived from other attributes.
- **SOCIO** compares the structures for the same sub-domain generated by different experts, or the same expert at different times or from varying perspectives.
- **EXPORT** formats the specifications of sub-domains as classes, of entities as objects, of attributes as properties, of entailments as methods, and of links as explanations, and transfers them to a performance tool, such as Object Nexpert (Rappaport 1986).

Figure 6 shows KSS0 in action eliciting knowledge about mapping techniques from a geographer as part of the development and validation of a geologic expert system (Shaw & Woodward 1987):

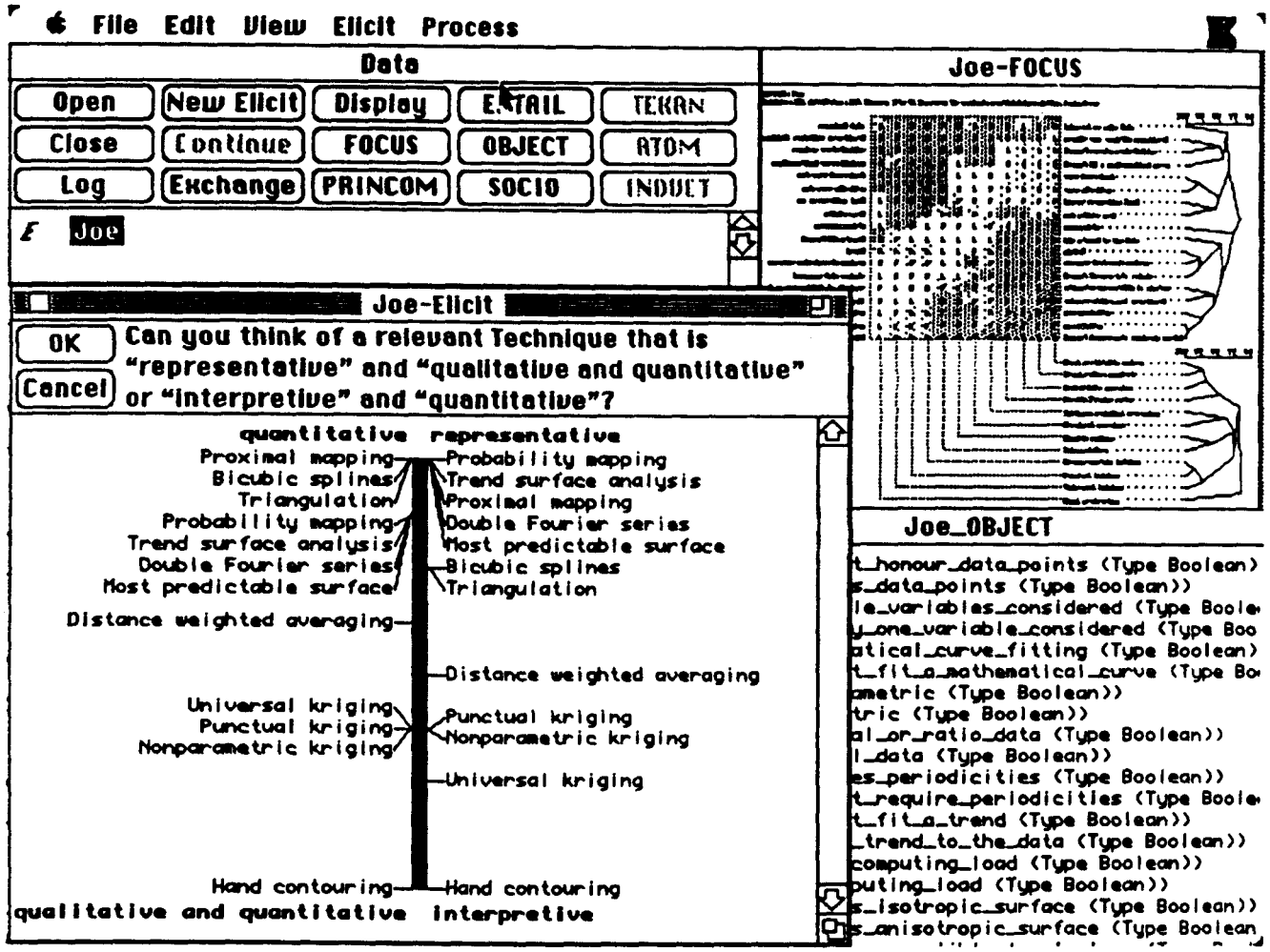

Fig.6 KSS0 in action

- The window at the top left is for session control giving access to the datasets and programs
- The window at the bottom left is for interactive entity and attribute elicitation. It is showing that two attributes are closely matched and prompting the expert to enter a new entity. All of the entities in the window are interactive—the expert can drag entities to new ratings on attributes, edit names, and so on.
- The window at the top right shows a hierarchical clustering of the entity-attribute data already elicited. This is used to elicit the object hierarchy from the expert.
- The window at the bottom right shows some of the properties, classes, objects and rules generated by the system for loading to Nexpert.

Figure 6 shows KSS0 in action eliciting knowledge about mapping techniques from a geographer as part of the development and validation of a geologic expert system (Shaw & Woodward 1987):

- The window at the top left is for session control giving access to the datasets and programs
- The window at the bottom left is for interactive entity and attribute elicitation. It is showing that two attributes are closely matched and prompting the expert to enter a new entity. All of the entities in the window are interactive—the expert can drag entities to new ratings on attributes, edit names, and so on.
- The window at the top right shows a hierarchical clustering of the entity-attribute data already elicited. This is used to elicit the object hierarchy from the expert.
- The window at the bottom right shows some of the properties, classes, objects and rules generated by the system for loading to Nexpert.

6 KSS0 as Automated Software Design

A simple example will show the way in which KSS0 acts as an automated software design tool. Figure 7 shows the entity-attribute representation of a set of data used by Cendrowska (1987) to demonstrate the inductive analysis of a set of ophthalmic data. The example is simple and the results well-defined which makes it a good test case. However, the requirements and results are sufficiently complex for the problem to be non-trivial and to demonstrate many major aspects of the object-oriented knowledge structures generated.

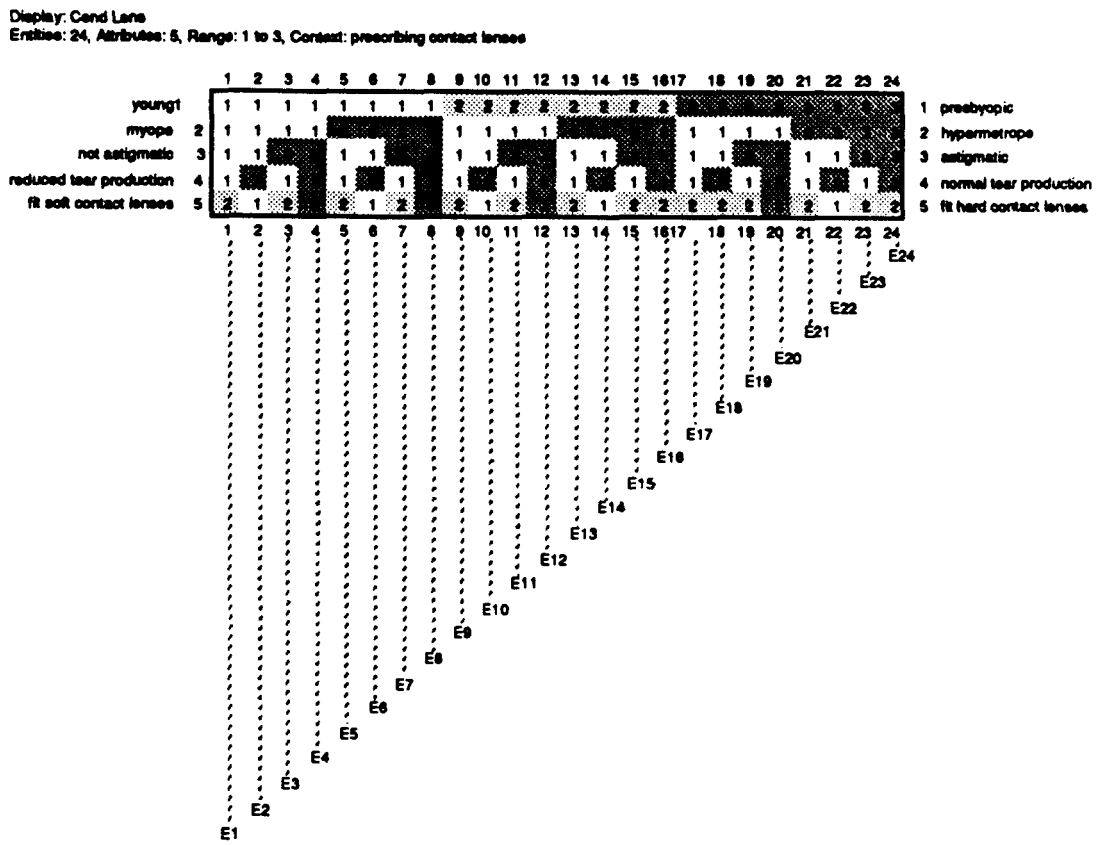

Fig.7 **Contact lens decision data in entity-attribute form**

The dataset in Figure 7 can be regarded as the definition of one class having 24 members (the cases) in terms of class attributes and the values of those attributes for each member. This class can itself be regarded as belonging to a superclass of datasets pertaining to the contact lens decision indexed by the expert whose knowledge is expressed. This is the basic class and object structure for the declarative knowledge involved.

Procedural knowledge can be derived from the dataset in Figure 7 by using inductive rule generation to link the descriptive attributes to the decision attribute. The resultant rule set (generated by the INDUCT algorithm, Gaines 1989) is shown in Figure 8. The first number after each rule is the percentage correct on the test data—it would be lower if the data were noisy. The second number is the statistical significance of the rule in terms of the percentage probability that it has arisen by chance.

```
prescription=hypermetrope & astigmatism=not astigmatic & tear
   production=normal -> lens recommendation=soft 100% 0.904%
age=young & astigmatism=not astigmatic & tear production=normal
   -> lens recommendation=soft 100% 4.34%
age=pre-presbyopic & astigmatism=not astigmatic & tear
   production=normal -> lens recommendation=soft 100% 4.34%

tear production=reduced -> lens recommendation=none 100%
   0.355%
age=presbyopic & prescription=myope & astigmatism=not astigmatic
   -> lens recommendation=none 100% 39.1%
age=pre-presbyopic & prescription=hypermetrope &
   astigmatism=astigmatic -> lens recommendation=none 100%
   39.1%
age=presbyopic & prescription=hypermetrope &
   astigmatism=astigmatic -> lens recommendation=none 100%
   39.1%

prescription=myope & astigmatism=astigmatic & tear
   production=normal -> lens recommendation=hard 100% 0.463%
age=young & astigmatism=astigmatic & tear production=normal ->
   lens recommendation=hard 100% 2.78%
```

Fig.8 Decision rules derived from the dataset of Figure 1

Figure 9 shows the automatic translation produced of this rule into a knowledge structure for an object-oriented shell. The first line defines the rule to be itself an object in the class of rules pertaining to the lens recommendation. This meta structure enables the use of rules within a context to be controlled using the natural data structures of the shell. The first two clauses in the rule also refer to meta-knowledge:

- The first clause tests the "use" field of the object "Cendrowska" (automatically defined as a member of the class of experts) to check whether the client wishes this expert's knowledge to influence the decision.
- The second clause tests whether a decision has already been made about the relevant hypothesis. This is used to implement default reasoning in which exceptional cases are tested first.

```
(@OBJECT=     lnd_6    (@CLASSES=
        lens_recommendation))
(@RULE=       lnd_6
        (@LHS=
                (Yes      (Cendrowska.use))
                (IsNot
        (<people>.lens_recommendation)   (KNOWN))
                (Is       (<people>.age)
        ("pre_presbyopic"))
                (Is       (<people>.prescription)
        ("hypermetrope"))
                (Is       (<people>.astigmatism)
        ("astigmatic"))
        )
        (@HYPO=       lnd_6.hypothesis)
        (@RHS=
                (Let
        (<people>.lens_recommendation)   ("none"))
        )
)
(@SLOT=       lnd_6.hypothesis          @INFCAT=
        10099;)
```

Fig.9 Translation of a rule into a knowledge structure for an object-oriented shell

The final line of the knowledge structure sets up the control structures for the default reasoning. The "inference category" attribute of the rule object is set to a value determined firstly by the probability that the rule is correct and secondly by the highest cost attribute tested (the cost of testing attributes is entered by the expert as meta-information). The overall result is that rules representing exceptions to the usual case are tested first, and, subject to this priority, rules testing expensive attributes are tested last, and only if necessary.

The net result of these structures is that the dataset of Figure 1 generates, without further intervention, a load-and-go expert system that performs exactly as required, testing the attributes in an appropriate order and not asking about the expensive attribute (tear-production) unless necessary. Additionally, the entire process is robust to noisy data and irrelevant attributes generating an operationally correct system even from very corrupt databases (Gaines 1989).

7 Conclusions

Developments in knowledge acquisition for knowledge-based systems development have led to a wide variety of techniques and tools being developed. The problem now is not so much to access research and experience in knowledge acquisition but to make sense of the diverse and wide ranging material available and, in particular, to apply the results to improve effectiveness in the development of knowledge-based systems. There are a number of major impediments to such understanding and application (Gaines 1987b):

- A diversity of techniques and tools that overlap in their applications and where it is not clear whether they are competitive alternatives or complementary partners
- Lack of variety, detail and evaluation in the case histories of applications of the techniques and tools
- Lack of access to the techniques and tools outside the narrow research communities originating them
- Lack of standardization in the knowledge representation resulting from the techniques and tools making it difficult to integrate them and to interface them to existing knowledge based systems
- Lack of standardization in the forms of data required by the techniques and tools making it difficult to apply them in the same situation and compare them
- Lack of standardization in the user interfaces to the interactive tools making it difficult to integrate them in an effective environment for human-computer interaction
- Lack of portability in the run-time environments required by the tools making it difficult to integrate them with other systems

These problems will not be unfamiliar to those involved in the development of software engineering techniques, tools and applications! It would be very useful to develop a substantial interchange between the knowledge acquisition and computer-aided software engineering communities.

From one perspective tools such as KSS0 are just CASE for a particular class of computing systems. Figure 4 suggests differences in emphasis and approach but the sources of information on the right hand side of this diagram are not peculiar to knowledge-based systems.

However, from another perspective the integration of "knowledge" into information-processing, with the connotations of subjectivity, uncertainty, disagreement and contradiction, introduces new problems of software engineering. We have developed methodologies for "hard" data that do not necessarily transfer easily to the "softer" data that is knowledge.

One philosophical definition of "knowledge" is "less than fact, more than opinion." It is this middle ground between the subjective and the objective that the new knowledge acquisition methodologies are treading.

These methodologies would benefit from the insights underlying modern computer-aided software engineering techniques. The interactive interviewing tools that have been developed for knowledge acquisition have applications to many aspects of system development. Hopefully, there will be increased interaction between the communities of researchers, tool developers, and applications developers involved in the knowledge acquisition and software engineering disciplines.

Acknowledgements

Financial assistance for this work has been made available by the Natural Sciences and Engineering Research Council of Canada. We are grateful to many colleagues at the knowledge acquisition workshops for discussions which have highlighted many of the issues raised in this paper. We are particularly grateful to John Boose, Jeff Bradshaw, Alain Rappaport and Brian Woodward for many fruitful discussions of knowledge acquisition systems.

References

Addis, T., Boose, J.H. & Gaines, B.R. (Eds) (1987) **Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge-Based Systems.** Reading, UK (September).

Boose, J.H. (1984). Personal construct theory and the transfer of human expertise. **Proceedings AAAI-84**, 27-33. California: American Association for Artificial Intelligence.

Boose, J.H. & Bradshaw, J.M. (1986) Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.5-0-5-16. Banff (November).

Boose, J.H. & Gaines, B.R. (Eds) (1986) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** Banff (November).

Boose, J.H. & Gaines, B.R. (Eds) (1987) **Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** Banff (October).

Boose, J.H. & Gaines, B.R. (Eds) (1988) **Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** Banff (October).

Boose, J.H. & Gaines, B.R. & Linster, M. (Eds) (1988) **Proceedings of the Second European Workshop on Knowledge Acquisition for Knowledge-Based Systems.** Bonn, Germany (June).

Boose, J.H. & Gaines, B.R. & Ganascia, J.G. (Eds) (1989) **Proceedings of the Third European Workshop on Knowledge Acquisition for Knowledge-Based Systems.** Paris, France (July).

Cendrowska, J. (1987) An algorithm for inducing modular rules. **International Journal of Man-Machine Studies** 27 (4), 349-370 (October).

Diederich, J., Ruhmann, I. & May, M. (1986) KRITON: A knowledge acquisition tool for expert systems. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.12-0-12-12. Banff (November).

Eshelman, L., Ehret, D., McDermott, J. & Tan, M. (1986) MOLE: A tenacious knowledge acquisition tool. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.13-0-13-12. Banff (November).

Gaines, B.R. (1986) An overview of knowledge acquisition and transfer. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.14-0-14-18. Banff (November).

Gaines, B.R. (1987a). Rapid prototyping for expert systems. Oliff, M., Ed. **Proceedings of International Conference on Expert Systems and the Leading Edge in Production Planning and Control.** pp.213-241. University of South Carolina.

Gaines, B.R. (1987b) Advanced expert system support environments. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.8-0-8-14. Banff (October).

Gaines, B.R. (1989) An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-Off between Expertise and Data based on Statistically Well-Founded Empirical Induction. **Proceedings of 6th International Workshop on Machine Learning,** Cornell University, Ithaca.

Hayes-Roth, F., Waterman, D.A. & Lenat, D.B., Eds. (1983). **Building Expert Systems.** Reading, Massachusetts: Addison-Wesley.

Johnson, N. (1987). Mediating representations in knowledge elicitation. Addis, T.R, Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge Based Systems.** pp.A2-1-10. Department of Computer Science, Reading University.

Klinker, G., Bentolila, J., Genetet, S., Grimes, M. & McDermott, J. (1986) KNACK—report-driven knowledge acquisition. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.23-0-23-13. Banff (November).

Marcus, S. (1986) Taking backtracking with a grain of SALT. Boose, J.H. & Gaines, B.R. (Eds) Proceedings of **the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.29-0-29-12. Banff (November).

Musen, M.A., Fagan, L.M., Combs, D.M. & Shortliffe, E.H. (1986) Using a domain model to drive an interactive knowledge editing tool. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.33-0-33-11. Banff (November).

Rappaport (1986). Multiple problem spaces in the knowledge design process. **Proceedings of AAAI Workshop on Knowledge Acquisition for Knowledge-Based Systems.** 37-0-37-15. Banff (November).

Shaw, M.L.G. & Gaines, B.R. (1983). A computer aid to knowledge engineering. **Proceedings of British Computer Society Conference on Expert Systems,** 263-271 (December). Cambridge.

Shaw, M.L.G. & Gaines, B.R. (1986) Techniques for knowledge acquisition and transfer. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the First AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.39-0-39-13. Banff (November).

Shaw, M.L.G. & Woodward, J.B. (1987) Validation of a knowledge support system. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Second AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.18-0-18-15. Banff (October).