Knowledge Acquisition: The current position.

Michael Wilson
Science and Engineering Research Council
Rutherford Appleton Laboratory

Introduction

Ten years ago it was stated that "...one of the greatest bottlenecks ... has been eliciting and programming new pieces of information... the theory does not exist in any sort of comprehensive codified form" (Buchanan et al, 1969). By 1983, the field had moved on so that one book could describe itself as marking "... the adolescence of a major new area of science and the birth of a new industry: knowledge engineering" (Hayes-Roth et al, 1983). At this time, "Forearmed with dire tales of the difficulties in store for them, knowledge engineers could be forgiven for embarking on the task of soliciting knowledge with all the optimism of a security interrogator extracting a confession from a diehard terrorist" (Cullen and Bryman, 1988). In this context several researchers attempted to identify a single "magical technique" which could be used to perform knowledge acquisition in all situations. Although some even claimed to have identified it they are generally assumed to be misguided.

More recently considerable interdisciplinary research between social and computer scientists has resulted in a range of methods which can be used for knowledge acquisition in different circumstances. The previous papers in this seminar, several recent books (Diaper, 1989; Greenwell, 1988; Kidd, 1987; Hart, 1989) and a recent substantial review paper (Neale, 1988) describe in depth the methods available for knowledge acquisition.

This paper is an attempt to supplement those sources by describing some of the current problem areas in knowledge acquisition and current research which addresses them. The general trend in knowledge acquisition research is to combine knowledge acquisition techniques with those from other areas such as software engineering and human computer interaction. From this position, the four research topics chosen for this review are : Multiple Knowledge Sources; Validating Acquired Knowledge; Software Development Methodologies and Process Models; Maintenance and Knowledge Base Extension.

Multiple Knowledge Sources

Standard approaches to knowledge acquisition describe many techniques which can be used with an expert to elicit knowledge. The use of a single individual as the expert in the design of an expert system is usually efficient when the system is required to problem solve in restricted domains or when a single, easily identified expert embodies most of the expertise the system is required to emulate. However, a single expert usually has expertise in only a small subset of tasks in a specific part of system and may not have adequate expertise in the other areas of the system's functionality. Indeed for many large systems, not only need multiple experts be used but also large sources of documentary data and possibly large stores of on-line information. Recent medical diagnosis systems have used the expertise from several specialists in different domains, text books, on line dictionaries and video libraries and large epidemiological data banks as sources of both the knowledge for the initial system and as a means of identifying changes in diagnostic data causing changes in the encoded knowledge. Such systems are remote in their design from the cognitive models underlying early expert systems and require elaborations on the simple elicitation techniques listed in most knowledge acquisition texts.

The simplest case where multiple knowledge sources are required is where there are several experts in non-overlapping domains of knowledge. Here knowledge elicitation can take place with the experts separately to acquire the static conceptual domain knowledge. If both that and the reasoning processes do not interact then knowledge elicitation can continue with the experts separately; since the final system will be modular in structure with clear data/knowledge interfaces. However, this is a rare case. Where the knowledge overlaps small-group knowledge acquisition should follow the initial individual sessions to determine that terms are being used in the same way and to clarify overlaps or contradictions. When conducting knowledge acquisition with groups, the usual problems of knowledge acquisition still exist with the addition of the social problems of managing a group of people (e.g. subordinates will not contradict superiors; some people won't make suggestions for fear of looking foolish to others; dictatorial aggressive individuals dominate the group etc ...). If the possibility exists to select the experts for group work, obviously the democratic individuals who will be motivated by the group's goals are better choices than self interested autocrats or non-directed individuals who may distract the group away from its objectives. However, such choices of experts are rarely available.

As when working with individuals, the social skills of the knowledge engineer are crucial to the enterprise, and training in group management skills is very useful. The major benefit of working with a group is that they will usually produce more options than an

individual. This is most pronounced in finding counter examples to an inference or in locating special cases. This characteristic can be amplified by conducting brainstorming sessions to generate ideas and then conducting consensus decision making to select which are the best or most appropriate. Otherwise, the techniques of task analysis, protocol analysis, structured interviews and simulation activities used with individual experts have proved useful (McGraw and Seale, 1988).

Apart from the use of multiple experts, the other sources of information required for most large knowledge bases are documentation or on-line data. The methods most often used for analysing and representing these are automated techniques of induction or machine learning which have been described in the papers by Forsyth and Addis.

Validating Acquired Knowledge

Conventional software is developed through methods which require documents specifying the user requirements against which designs and programs can be validated to show that they perform the function required. Such conventional validation requires precise test procedures. As long as reasonably precise requirements and design specifications can be obtained for KBS, test procedure preparation should be of no greater difficulty than for conventional software. Although the KBS development methodologies discussed in the next section, and recent work on quality assurance for KBS (Born, 1988) attempt to define the required documents, KBS software requirements are often nonexistent, imprecise, or rapidly changing. In this situation test procedure design becomes more difficult.

There is no widely accepted, reliable method for evaluating the results of tests of expert systems. Rushby (1988) proposes that evaluatable requirements can be divided into the "service" and "competency" requirements. Service requirements are such things as the input and output formats expected, processing rates, the explanation facilities required and other factors which should be amenable to statements no less rigorous and formal than those for conventional software. Competency requirements concern the knowledge acquired for the system and can be divided into minimum and desired competency requirements. The minimum competency requirements should define how badly the system is allowed to perform. These may be similar to system safety specifications and should be reasonably easy to specify. Desired competency will probably be defined relative to human expertise and describes how well a system should perform, which will be more difficult to specify. For example, the desired competency of a system may be to produce an optimal value which may be hard to find, and its optimality just as hard to check; it may be easier to define a minimum requirement that the system produce a valid solution. The usual method for defining desired competency is through a gold standard.

This will either be an objectively correct answer to a problem, or an answer given by a human expert (or group of experts) when presented with the same information as the system being evaluated. However, the approach of having human experts in the domain of the expert system evaluate the results has numerous drawbacks: there may be no expert available, or the expert may not be independent when independent evaluation is required; human experts may be prejudiced or parochial; the problem for which the expert system was written may be one that no human can solve reliably or efficiently. Evaluation using human experts is expensive, so few test cases are generally used with the consequential failure in the completeness of the evaluation (the medical diagnosis system MYCIN was subjected to a mere 10 test cases, and the configuration system R1 only 50 before being put into service).

As with conventional software, KBS can be subjected to two forms of testing to locate potential problems in the knowledge acquisition: static and dynamic testing. The most common forms of static testing involve experts checking for consistency and completeness in the conceptual knowledge of a domain (usually on a paper model of the knowledge base), and identifying critical values and cases for dynamic testing. A minimum requirement for dynamic testing is that every rule should be made to fire once, and a more stringent test would be that all outcomes of every rule should be exercised. Several research projects are currently investigating the notion of measures for testing combinations of rules and outcomes in the spirit of path testing for conventional software. However, the only common methods of dynamic testing for KBS are to test critical values of rule combinations which are known to change the dynamic operation of the system catastrophically. Along with this it is often useful to perform a sensitivity analysis to determine both, if the same input can produce different outputs and whether very similar inputs can produce wildly different outputs - it is this potential for instability that underlies many of the concerns expressed about AI software. It is not the number of test cases used which is of importance (although this does give face validity to any evaluation), but the coverage of the test cases. These should cover a random sample of the possible data combinations (not just the easy or common cases), and additional obscure or complex test cases which experts find difficult.

Validation and testing should not be thought of as extraneous to knowledge acquisition, since it is these methods which will motivate the knowledge acquisition in its later stages to refine the knowledge base.

Software Development Methodologies and Process Models

So far knowledge acquisition or knowledge elicitation have not been defined in this paper. Kidd (1987) suggested a simple definition which is similar to that used by Diaper in the opening paper of this seminar: "Knowledge acquisition involves elicitation of data from the expert (usually by some verbal technique); interpretation of the data to infer the underlying knowledge or reasoning process, and, guided by this interpretation, creation of a model of the expert's domain knowledge and performance". Although this is adequate as a definition, it is not a method for performing knowledge acquisition or KBS development. Equally, although the AI literature abounds with methods and techniques for eliciting knowledge and modelling reasoning processes, their description is not uniform and unambiguous. Consequently, a major research topic at present is that of integrating these techniques and decisions made about knowledge acquisition with the rest of the software development method.

Figure 1: A Knowledge Based System development life cycle.

There are many development methods for non-KBS software (that do not include knowledge acquisition) based on development or product life cycles. As a result of inter-disciplinary research between social and computer scientists several of the major corporations that are now using KBS have established their own methods of the style used for non-KBS software which are aimed at KBS development. Figure 1 shows a schematic of one method proposed by Tectronics.

This is a high level view of a software development process which allows the techniques and decision points to be fitted into it. At a finer grain of description it allows the interaction of decisions made at different stages to be seen; for example, the interaction of decisions about the choice of suitable KBS applications with those made during knowledge elicitation and representation.

The more detailed description of the problem definition stage includes the option for a complete cycle of the method on a part of the identified problem as a feasibility study. This includes the development of a prototype which is often required as much to show the client what a KBS could do and persuade them to fund further development as it is for knowledge validation, human interface evaluation or evolutionary system design.

The definition of the problem and the assessment of KBS as a suitable solution to a problem is itself still a major research issue. Prerau (1985), Waterman (1986) and Slagle and Wick (1988) all suggest criteria for this assessment, although they are not complete or systematically integrated into a broader development method. Slagle and Wick suggest the most complex criteria as lists of essential and desirable features of a potential application (shown in table 1). A score between 0 and 10 should be assigned as a general weight of importance to each feature in the lists, and then as a value for each candidate application. The sum of the products of the weight and value shows which are the most likely applications (their paper should be consulted for the complete method before it is attempted). Greenwell (1988) presents similar lists of questions for use in the first knowledge acquisition sessions which do not rely on complex weighting schemes to evaluate the suitability of applications. One drawback with these lists of criteria is that they are implicitly linked to the knowledge elicitation, analysis and representation techniques used by the authors. For example, one of Slagle and Wick's essential features for an application is that the "expert does not use physical skills". It is true that expert's are unlikely to be able to verbalise the details of how they use physical skills and the development of a KBS using interview or other verbal knowledge elicitation procedures would be impossible in these cases. However, task analysis of records of performance, or induction of rules from data acquired through transducers on the moved devices (e.g. readings from meters read or values from knobs moved in a control plant) will give

information which can be included in a KBS.

Essential Features	Desirable Features
Recipients agree on high payoff	Management committed to follow on
Recipients have realistic expectations	Insertion into work place smooth
Project has management commitment	System interacts with user
Task is not natural language intensive	System can explain reasoning
Task is knowledge intensive	System can intelligently question user
Test cases are available	Task identified as problem area
Incremental growth is possible	Solutions are expandable
Task requires no common sense	Task does not require real-time response
Task does not require optimal solution	Similar expert systems built before
Task will be performed in the future	Task preformed in many locations
Task not essential to deadline	Task performed in hostile environment
Task easy, but not too easy	Task involves subjective factors
An expert exists	Expert unavailable in future
Expert is a genuine expert	Expert intellectually attached to project
Expert is committed to entire project	Expert does not feel threatened
Expert is co-operative	Expertise loosely organised
Expert is articulate	
Expert has successful history	
Expert uses symbolic reasoning	
Hard to transfer expertise	
Expert does not use physical skills	
Experts agree on good solutions	
Expert does not need creativity	

Table 1: Essential and Desirable features for a KBS application (after Slagle and Wick, 1988).

For several years there has been an unfruitful debate about which single development method should be followed: a linear waterfall model, an evolutionary approach, should prototyping be included or only static paper representations, etc. Many of the corporate KBS development methods such as that in Figure 1 are based on a linear waterfall model which has been slightly modified to accommodate the prototyping used in KBS development. Recent research in software engineering has moved away from this linear approach of the life cycle in order to take account of dependencies between decisions taken in different parts of the development (e.g. the selection of suitable applications and the selection of knowledge acquisition techniques), and to allow methods to be selected during development which are appropriate to address problems that arise. The most advanced approach is the spiral model of software development and enhancement proposed by Boehm(1988). This allows the method to be applied between reviews which most addresses the aspect of the development where the greatest risk of the project failing will come from (e.g. changes in the user requirements, the human interface, data interfaces between modules, the completeness and consistency of static knowledge, or the dynamic interaction of knowledge and inference processes). The major research

project on KBS methodology is the CEC funded project KADS (Knowledge Acquisition and Design System, see Hayward, 1987) that started in 1985 and will continue until 1990. Early in the KADS project they proposed a waterfall style life cycle model which did not include prototyping, although as research has progressed they have shown the need to represent the links between decisions at different stages of development and are now investigating a model of the spiral class.

The KADS development method addresses in detail the interaction of knowledge acquisition with the organisational structure and strategy of the user organisation; the integration of user requirements themselves with the knowledge acquisition; the human interface design and other human factors issues of the system including training; as well as providing tools to support the use of the overall method. There is not sufficient space here to describe all these developments, although it should be noted that knowledge acquisition will interact with many other issues, and they should all be considered. The knowledge acquisition stage in the KADS model however includes sufficient useful and portable features to require a brief summary.

KADS proposes that knowledge modelling should pass through five stages:

1) Data - a low level description which can be mapped onto linguistic information.

2) Conceptual Model - describes the competence in expert problem solving

3) Design Model - high level system design at the same abstraction as the Conceptual model although including implementation formalisms.

4) Detailed Design Model - a transformation from the design model

5) Implementation - the implemented system.

The focus of knowledge acquisition within this methodology will be the data and the conceptual model resulting from its analysis and representation. The Conceptual Model is further divided into a four layer knowledge model where each successive layer interprets the description of the lower layer:

1) static domain knowledge - domain concepts and their attributes, domain facts, structures representing complex relations etc .. The static domain knowledge is assumed to be largely class neutral.

2) knowledge sources - elementary steps in reasoning that specify the type of information used in making the inference and the type of information that it produces.

3) task - the structure of elementary inferences required to produce a goal.

4) strategic knowledge - which goals are relevant to solve a particular problem.

To facilitate the use of this approach the KADS project has proposed a formal notation for specifying conceptual models which can be used to analyse data. Perhaps the major contribution of the project has been to specify a set of generic interpretation models for different problems which act as skeletons for conceptual models. In table 2, one of the desirable properties of a suitable KBS application was that similar expert systems had been built before. The experience of building a similar system provides the knowledge engineer with a framework of what classes of knowledge to look for, what inference processes to expect and what elicitation and representation techniques are suitable. The interpretation models proposed in the KADS project serve the role of this prior experience in that once the class of problem has been identified, then they identify the classes of information to look for in the conceptual model. Although knowledge engineers may not use the notation proposed in KADS, the use of interpretation models is a valuable and useful technique in motivating the structure of the knowledge acquisition process. It is always easier to look for something when you know what it is that you are looking for, then when you do not. Knowledge engineers may not even wish to use the interpretation models proposed within KADS since they adopt the four level model of knowledge. However, the construction of interpretation models to fit other knowledge abstractions based on personal experience would serve the same purpose.

Having stated the desirability and usefulness of interpretation models it is necessary to state some of the failings and problems with them. It is tempting when using them to stick too closely to them, or to the initial selection of an inappropriate model. This will cause the knowledge engineer to look for knowledge which is neither relevant or present, or to overlook some of the knowledge which is presented. A second failing is limited to the current KADS models, in that they assume that all knowledge will be acquired verbally. Consequently, they do not include guides as to which knowledge elicitation techniques should be used of the sort which Welbank describes in her paper. These would be a valuable extension to the interpretation models and may be included later in the project. There are three main reasons for this omission. Firstly, although KADS proposes a set of knowledge types for each of the layers of the conceptual model, their is uncertainty as to the correctness and completeness of these. Secondly, elicitation method selection tables are normally built up from studies conducted by various authors who use disparate and often incompatible knowledge types. This can result in confusion when trying to select techniques. If the knowledge types are mapped onto a single theoretical base such as that used in KADS, then this problem could be overcome. Thirdly, the theoretical basis in cognitive psychology and epistemology of the knowledge types and approach of KADS is grounded on verbal concepts, and there is little theoretical support for including

concepts which have no verbal status.

The importance of development methods for knowledge acquisition is that they allow different techniques, guidelines and decision points to be clearly stated in a coherent whole which can guide the knowledge engineer, without him having to resort to a jumble of incompatible and inconsistently expressed case studies or academic papers. This allows the interaction of decisions at very distant parts of the development method to be traced, and the consequences of any decision made clear to the knowledge engineer.

Maintenance and Knowledge Base Extension

Although maintenance and enhancement is clearly a stage in the software process model, it has been classed as a separate research topic since it is often overlooked in discussions of knowledge acquisition. The major difficulty with advice on maintenance is that there is very little data or published experience on maintaining KBS. Because of this, the main points to be made about maintaining KBS are to note potential problems. The most important observation is that knowledge acquisition is not completed when a system is delivered to a customer, but continues during maintenance and extension of the system. For example, XCON is a rule based system used by DEC to configure computer systems since January 1980. Over 7 years it grew from 700 to 6,200 rules of which 50% changed every year (these draw on a database of approximately 20,000 parts so the changes are not simply those required to add new parts to be configured).

Two main properties directly influence the maintainability of a KBS. The homogeneity of the maintained representation: that is the use of a small number of discernible representations over and over again to accomplish the various desired goals; and the predictability of the representation: that is, readers know where to look for the answers to questions; they are not surprised by what they come across; and they can trust that nothing is hidden. The term representation is used here, since the object which is being maintained as new knowledge is acquired may not always be code, but an intermediate representation. The use of an intermediate representation will add an extra layer to the maintenance, but it will also provide a higher level language in which these properties are more likely to persist.

One of the problems in maintenance is that it is rarely performed by the original knowledge engineer; often not by a member of the team or organisation which developed the original system; and in some cases (e.g. for changes to a financial system immediately after a budget announcement) by the client or user under time pressure. It is equally common that expertise will have to be acquired from experts who were not involved in the original knowledge acquisition and may have different ways of describing the domain

than those who were. These changes in personnel require that the original knowledge representation and the knowledge acquisition process used are carefully documented so that a new knowledge engineer can understand them. A typical strategy for extending a representation to handle new examples of prior static knowledge is to copy the representation that worked for a similar body of static knowledge and then edit it to handle the new knowledge. Unfortunately, in the editing process one is not always sure what the rationale for all the functions was. The result is that one often inadvertently changes a function; alternatively, one doesn't change the functions, but keeps them in the new representation - not feeling confident why they were there. The rationale behind the representation of the knowledge should be clearly documented to avoid these problems. The problems of maintaining KBS and knowledge acquisition during maintenance are greater than those in non-KBS programs - there has been an unfortunate tendency to regard KBS as different and forget the lessons learned elsewhere.

There is no clear solution to the problems of maintenance. It is necessary to consider it when performing the initial knowledge acquisition and to document all the decisions made and techniques used carefully to reduce problems when it finally comes about. One useful test of representations and documentation is to see if it is necessary to know who wrote them before they can be understood (or to see if the person who wrote them can be identified from them) - if so, they are inadequate.

Conclusion

This paper has attempted to cover the major areas of current knowledge acquisition research which the other papers in this seminar or the available books on the field do not address. I apologise if some information is duplicated between this paper and those of the other speakers, or if I assumed another speaker would cover issues that have been missed since they did not. The paper does not address the core issues of knowledge elicitation or representation techniques which occupy most pages in descriptions of knowledge acquisition, and consequently may read as though it is peripheral to the subject. However, the issues of multiple sources of knowledge, validation of knowledge, development methodologies, the selection of appropriate applications and maintenance which have been addressed all bear on decisions made during knowledge acquisition and if not performed as part of the knowledge acquisition process, must be considered during it. Unfortunately, most of the issues mentioned have not been resolved since they are still problematic, and the subject of research. The pointers that are available from that research provide as much guidance as is available.

References

Boehm, B.W. (1988), "A Spiral Model of Software Development and Enhancement", *Computer*, May 1988, pp 61-72.

Born, G. (1988), *Guidelines for the Quality Assurance of Expert Systems.* London, U.K.: Computing Services Association.

Buchanan, B.G., Sutherland, G.L. and Feigenbaum, E.A. (1969), "Rediscovering some problems in Artificial Intelligence in the Context of Organic Chemistry". In B Meltzr and D. Michie (eds) *Machine Intelligence*, **5,** pp 253-280, Edinburgh: Edinburgh University Press.

Cullen, J. and Bryman, A. (1988), "The Knowledge Acquisition Bottleneck: Time for Reassessment?" *Expert Systems,* **5** (3), 216-225.

Diaper, D. (1989), *Knowledge Elicitation: principles, techniques and applications.* Chichester, U.K.: Ellis Horwood.

Greenwell, (1988), *Knowledge Engineering for Expert Systems.* Chichester, U.K.: Ellis Horwood.

Hart, A. (1989) *Knowledge Acquisition for Expert Systems (Second Edition)* London, UK: Kogan Page

Hayward, S., (1987), "How to build knowledge based systems: techniques, tools, and case studies". In *ESPRIT '87: Proceedings of the Esprit Conference*, pp 665-687. Brussels: Commission of the European Economic Community.

Hayes-Roth, F., Waterman, D.A., and Lenet, D.B., (1983), *Building Expert Systems,* Reading, Mass.: Addison-Wesley.

Kidd, A., (1987), *Knowledge Acquisition for Expert Systems: A Practical Handbook.* New York, USA: Plenum Press.

McGraw, K.L. and Seale, M.R., (1988), "Knowledge Elicitation with Multiple Experts: Considerations and Techniques" *Artificial Intelligence Review*, **2** (1), pp 31-44.

Neale, I.M., (1988), "First Generation expert systems: a review of knowledge acquisition methodologies" *The Knowledge Engineering Review,* **3** (2), pp 105-145.

Prerau, D.S., (1985), "Selecting an Appropriate Domain for an Expert System" *AI Magazine,* **6** (2), Summer, pp 26-30.

Rushby, J. (1988) *Quality Measures and Assurance for AI software,* SRI International Report, Sept. 1988.

Slagle, J.R. and Wick, M.R. (1988), "A method for evaluating candidate expert system applications" *AI Magazine,* **9** (4), Winter, pp 44-53.