Watson, I.D., Basden, A., & Brandon, P.S., (1992). A Client Centred Approach to the Maintenance of Expert Systems. Expert Systems, Vol. 9 (iv): pp189-96.

A Client Centred Approach to the Maintenance of: Expert Systems

Ian Watson*, Andrew Basden† & Peter Brandon*

*The EDESIRL Project
Department of Surveying
University of Salford
Salford, M5 4WT, UK.
☎ +44 61-295-5227 🖹 +44 61-295-5011
💻 I.D.WATSON@CONSTRUCT-IT.SALFORD.AC.UK

†Information Technology Institute
University of Salford
Salford, M5 4WT, UK.

ABSTRACT

Developing an expert system is a considerable investment in time and money, and yet very little attention has focused on keeping expert systems in regular beneficial use. The problems of maintaining expert systems, and the lack of research this area has attracted is a major contributing factor to the continued lack of acceptance of these systems. This paper discusses the maintenance of expert systems within the context of the Client Centred Approach. Although there are many similarities between maintaining expert systems and conventional systems maintenance there is one vital difference — **knowledge changes**. Consequently an expert system requires regular evolutionary maintenance if its performance is not to be impaired.

This paper is intended to promote discussion of maintenance issues within the community. We review some principles and techniques that can assist the maintenance of expert systems and describe some formative work towards a maintenance methodology derived from a case study of maintaining a commercially available expert system.

INTRODUCTION

Over the last twenty years there has been a considerable amount of successful research into many aspects of the development of expert systems (ESs). This has dealt with logic programming techniques, knowledge representations, knowledge elicitation and acquisition methods, development tool kits, and more recently development methodologies. Despite all the successes, ESs have still not realised a fraction of their potential, particularly within

smaller organisations. We argue that problems of maintaining ESs in regular beneficial use are major contributing factors to the slow uptake of ESs [Coenen, 1992].

In comparison to the quantity of research into other aspects of ESs, maintenance has received relatively little attention. This may be due to the early, and often repeated claims that the modularity of production rules makes maintenance "simple" [Shortliffe, 1976; Barr & Feigenbaum, 1982]. However, the experience of R1/XCON has clearly demonstrated that maintenance is not simple [Bachant & McDermott, I984; Solway et al., 1987].

This paper identifies continual changes in knowledge as the main problem of maintaining ESs. We also question the opinion that modular rules are easy to maintain, and we recommend that knowledge bases are maintained at the knowledge level [Newell 1982].

The paper opens by outlining the Client Centred Approach (CCA). This is an ES development methodology designed for small and medium sized enterprises (SMEs). It explicitly identifies that maintenance is a major stage in the life cycle of a system and thus must be planned. The principles of the CCA are described in Basden [1989]. Its current state of development is detailed in Watson et al. [1992]. We then review some maintenance techniques, including common sense ones, that can assist maintainers.

The paper concludes with the results and analysis of a case study of an update to a commercially available ES. This study has helped us formulate a maintenance methodology that is grounded in commercial practice rather than laboratory experience.

THE CLIENT CENTRED APPROACH

The Background

Basden [1989] argues that a problem common to most current ES development methods is that they are "technology centred". They place too much emphasis on the activities used to develop systems, such as "elicitation," "implementation," and "verification" and not enough emphasis on what clients can see and understand. It has been argued that by putting people at the centre of the development process [Diaper, 1987 & 1989] there is a greater chance of the resulting system being useful.

Basden, however, identifies the "client" as an individual or group distinct from the eventual end users of the system. The concept of the client represents all stakeholders in the system and considers

- the motivation for building the ES,
- the provision of resources for building the ES (including domain expertise),
- the plan for building and using the system, and

- the responsibility for the impact of the ES when in use.

The Client Centred Approach (CCA) is being developed specifically for SMEs and is the subject of a three year collaborative research project at the University of Salford. The CCA covers the full development life cycle of an ES providing milestones or deliverables to guide the project. These refer to what the clients can see being demonstrated and not to conventional tasks of elicitation, acquisition, and so forth. This accepts that clients may not understand the jargon or the distinction between tasks involved in development but that they will be able to perceive demonstrable changes in the system.

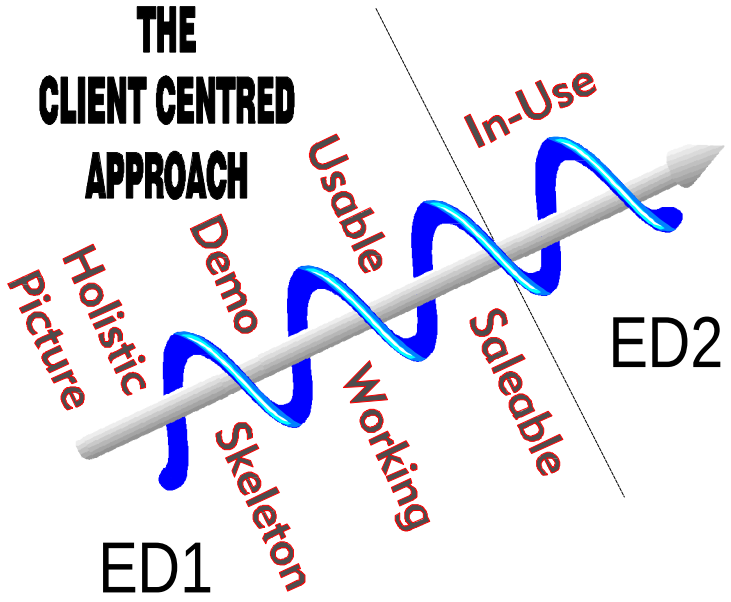

Figure 1. Seven Stages of the CCA

An Overview of the CCA

The stages of the CCA are illustrated in Figure 1 and are described in more detail in Watson et al., [1992]. The CCA is divided into two broad activities:

- **Evolutionary development part one (ED1).** This considers the development of the ES and takes it to a saleable stage.
- **Evolutionary development part two (ED2).** This ensures that the system is used correctly by checking that the clients, users, and their organisations understand the strengths and weaknesses of the ES. It also involves maintaining the knowledge base and regularly updating the functionality of the system. This ensures that the ES remains in beneficial use over the longest possible time, thereby maximising the return on the investment in its development. Hicks [1990] argues that maintenance must be made an

explicit part of any ES development plan. When an application is brought into productive use, the program must be maintained, and therefore resources have to be planned for and allocated. ED2 within the CCA directly addresses this issue.

MAINTAINING EXPERT SYSTEMS

ESs are characterised by the volume of explicit knowledge they contain. This knowledge changes over time, and in some instances quite rapidly. For example a KBS developed for British Coal's Insurance and Pensions Division requires an average of two changes per week due to changes in legislation, legal judgements, and company policy [Bench-Capon & Coenen, 1992a]. This implies that if an ES is not maintainable it may be out of date almost before it is installed [Bratley, et al., 1991; Bench-Capon & Coenen, 1992b]

ESs must therefore be designed for easy maintenance. This paper does not propose any easy solutions for maintaining ESs. However, poor practice during development will make maintenance difficult or even impossible. We accept that the maintenance of any computer software should involve "common sense" and good programming principles such as

- remember that developers may not be responsible for maintaining the system;
- record **why** the original design decisions where taken;
- comment code adequately during development;
- identify sources of knowledge (i.e., who supplied a piece of knowledge, or where it was obtained);
- modularise the system either by knowledge chunks or by functionality;
- use data flow diagrams to describe the inputs and outputs of functional modules [Prerau et al., 1990];
- keep antecedents and consequents of rules simple;
- keep code simple and easy to read [Prerau et al., 1990];
- use an intermediate representation to model the knowledge [Berry & Broadbent, 1986; Johnson, 1989; Watson et al., 1989; Bench-Capon & Coenen, 1992b; Coenen & Bench-Capon, 1992; Kalos, 1992]; and
- record why modifications were made, who made them, and when they were made.

Of the above common sense advice, recording "why" decisions were taken is perhaps the easiest implemented, and from our experience vitally import. The program code itself is the definitive statement of "what" was done, and it can be read easily by an experienced programmer. The maintainer needs to know "why" decisions were made: "Why are all the variable names eight characters long?" "Why is that calculation performed by an outside routine?" and "Why is that question always asked first?" Implementing this, and following the advice above, will not necessarily reduce system maintenance, but it will certainly make it easier.

Conventional Software vs. Expert Systems

While constructing an ES is similar to constructing any other piece of software its maintenance is fundamentally different [Chee & Power, 1990; Pau & Kristinsson, 1989, Killin et al., 1991]. For example, conventional software maintenance involves the following tasks:

Adaptive Maintenance — Involves adapting existing code to new environments (e.g., porting a system from DOS to UNIX). Note that we are using a more restrictive definition of adaptive maintenance than that proposed by Swanson [1976].

Corrective Maintenance — This involves correcting errors in system performance by testing results against known data and validating it with experts. This corrective maintenance is likely to be more difficult for ESs than for conventional systems since there are likely to be fewer of the benefits of modular code and testable units.

Perfective Maintenance — This involves enhancing system functionality and includes usability issues. Perfective maintenance is different from corrective maintenance since it involves improvements to the system, not corrections.

All of these types of maintenance may be found with conventional systems. However, as described above, the maintenance of ESs involves a problem less often found in conventional systems

Evolutionary Maintenance — The knowledge in the ES does not remain static over time. Working practice or legislation may change, or new "better" knowledge may become available. Thus, designers of ESs must plan for inevitable changes in domain knowledge [Morris, 1989; Hicks, 1990; Bratley et al., 1991; Bench-Capon & Coenen, 1992a]. Moreover, unlike some conventional software it is almost impossible to predict and plan post-delivery maintenance [Caviedes, 1987]. Even if the system were one hundred per cent correct and complete (i.e., perfect) that state is only transitory, and over time the system will require evolutionary changes to its knowledge base.

KNOWLEDGE BASE METRICS

To maintain an ES one needs to know which elements require maintenance. Recording bugs, and changes to knowledge are routine activities that inform the maintenance requirements. However, some researchers have proposed using knowledge base metrics to assist maintainers in identifying problem areas. Examples of these metrics are described below.

Knowledge Utilisation

An ES may be used to provide routine advice ninety per cent of the time and unforeseen exceptional advice ten per cent of the time. It is on the boundaries of their expertise that ESs experience fragility [Hart, 1988]. It is therefore useful for system maintainers to record the utilisation of knowledge [Madeo & Levary, 1990].

Each rule included in an ES contributes to the overall characteristics of the system. For example, it can happen that some rules will be entirely unused. These rules may be redundant. Other rules may be employed excessively, which may suggest a need for further refinement such as expansion [Madeo & Levary, 1990]. The utilisation of each rule in an ES can be easily logged by counters indicating the number of times each rule fires.

Knowledge Stability

Since knowledge changes over time it is reasonable to assume that knowledge that survives in a knowledge base for some time without any modification is validated to a greater degree than more recent knowledge [Bentley et al., 1992]. Bentley et al., recommend recording the age of rules in a knowledge base along with recording

- the oldest rule(s),
- the youngest rule(s),
- the mean age of rules to be used as an indicator of the overall confidence in the acceptability of versions of the ES, and
- the standard deviation from the mean age used as an indicator of the relative acceptability of rules in the knowledge base.

Confidence

Bentley et al., [1992] recommend that knowledge persistence should be combined with usage statistics to form a confidence factor that offers integrated data to assist the maintenance and validation of ESs. If the confidence factor of a rule indicates that it is old and never used, knowledge engineers should investigate it. They must then decide if the unused rule is misstated, if the rule is redundant, or if there is a problem elsewhere in the system that causes the rule not to fire. The knowledge engineers may have to decide whether the unused rule will ever serve any useful purpose [Madeo & Levary, 1990; Bentley et al., 1992].

KNOWLEDGE LEVEL MAINTENANCE

It is often assumed that maintaining an ES is easier than maintaining conventional systems because the explicit separation of knowledge from control in an ES helps maintainers make and monitor changes [Shortliffe, 1976; Bar & Feigenbaum, 1982]. The mechanistic ease with which rules can be modified, added or deleted is usually quoted as proof that ESs are easy to maintain. There is some evidence to support this view. For example, the Fraudwatch system developed for Barclaycard as an experiment within the KADS project, requires about eight weeks work every six months to maintain [Killin et al., 1991]. Evidence that we present below also shows a similar level of effort to maintain a commercial ES.

However, the lesson of XCON teaches that maintaining an ES is not always simple [Bachant & McDermott, 1984]. Modifications to rules are not isolated, and predicting the effects of changes in a large knowledge base are very difficult [Bench-Capon & Coenen, 1991]. A popular solution to this problem is the use of knowledge base debugging tools.

There are a plethora of these for almost every conceivable development language, environment and knowledge representation, and there would not be the space to cite them here. However most of them have been developed to aid the development of ESs and are in essence debugging aids since they operate on executable code (i.e., at Newell's symbol level [Newell, 1982]).

In contrast, we suggest that evolutionary maintenance should focus on the knowledge level [Newell, 1982], since it is the knowledge in the system that is evolving. Changes to the system's code are merely a consequence of this [Bench-Capon & Coenen, 1992b]. In the section on "common sense" we advised that ES developers should use intermediate knowledge representations to model the knowledge in the system.

The advantages of using intermediate representations in the development of ESs

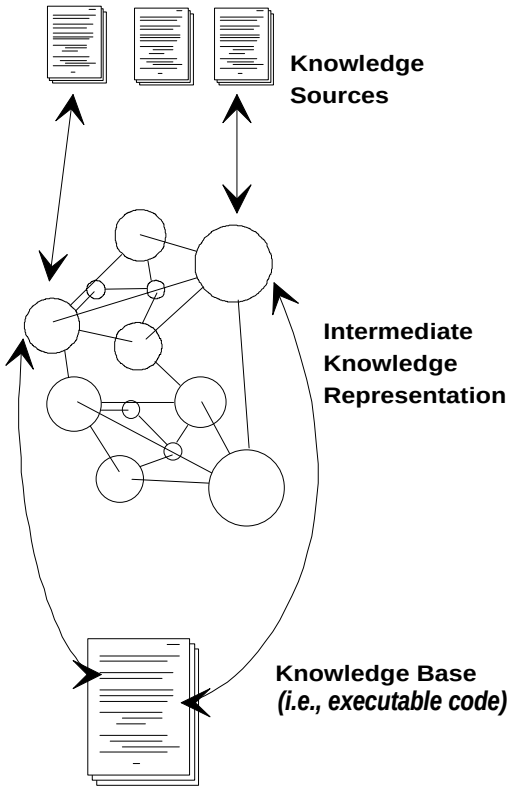

Figure 2. Mediating Representations

are often cited [Alexander et al., 1986; Wielinga & Breuker, 1986; Berry & Broadbent, 1986; Butler & Chamberlin, 1987; Edwards, 1987; Johnson, 1989; Recogzei & Plantinga, 1987; Young, 1987; Diaper, 1987, Watson et al., 1989; Kalos, 1992; Watson et al., 1992]. However, there are also clear benefits for maintenance [Bench-Capon & Coenen, 1992a & 1992b]:

- maintenance can be carried out on the intermediate representation and then verified, and
- an isomorphism can be established between the knowledge base and the knowledge sources, with the intermediate representation "mediating" between the two [Johnson, 1989; Bench-Capon & Coenen, 1992b]. The position of a mediating representation (i.e., between the knowledge and the code) is shown in Figure 2.

The task of maintaining or editing an intermediate knowledge representation need not be unsupported by tools [Motta, et al., 1989, Brooke & Jackson, 1991; Watson & Norman, 1992]. Indeed, verifying an intermediate representation manually is likely to be extremely hard without software support.

The basic functionality of a tool for verifying intermediate representations should test for

- conflicts or contradictions between elements of the representation;
- redundant elements;
- inconsistent elements;
- logical loops in the representation; and
- isolates, i.e., elements that are not connected to the knowledge base and could never be invoked by the knowledge base.

The tool should also help locate knowledge sources and their corresponding elements in the knowledge base. The implementation of a tool for testing inference nets (the intermediate representation our project uses) is described in Watson & Norman [1992], whilst Bench-Capon & Coenen [1991] describe a similar tool for maintaining KANT structures.

A MAINTENANCE CASE STUDY

One of our project collaborators, Imaginor Systems, develops and sells ESs for the construction industry. We have closely monitored their latest update to one of their systems and are offering the results as quantitative evidence of the effort involved in maintaining an ES. This study has also helped us develop a maintenance methodology grounded in commercial practice as opposed to research laboratory experiments.

The ELSIE ES was developed by an Alvey Community Club Project [Brandon et al., 1988]. The system estimates the cost and development time for commercial office developments. It also advises on the best procurement method for the development and on the likely returns on investment (i.e., development appraisal). The ES has been commercially available since 1988 and has sold over 400 units. It is used by surveyors, architects and other construction professionals throughout the UK. We monitored the update from version 4 to version 5 of ELSIE. The work was carried out by a single person over a period of eight weeks.

The maintainer recorded a daily log of their activities; from this, several results have emerged. Table 1 shows a breakdown of the time spent on various activities. Note that the maintenance stage proper has been broken down into corrective, perfective, and evolutionary maintenance.

The figures from our study show that maintainers of an ES should allow as much time for the planning, preparation, familiarisation, and house keeping tasks as for the actual maintenance itself. The time spent on corrective maintenance for this update was low indicating that ELSIE Version 4 was a mature and stable system (i.e., most of the bugs have already been dealt with). Maintainers should therefore expect to spend considerably more time on corrective maintenance for a younger system.

Maintenance Activities	**Time** (in hours)
Preparing prioritised list of changes to ES	20
Familiarisation with existing ES	4
Preparing ES for maintenance (backing up ES, etc.)	35
Corrective Maintenance	22
Perfective Maintenance	149
Evolutionary Maintenance	68
House Keeping Duties (version control, documentation)	84
TOTAL	382

Table 1. Breakdown of ELSIE Update Effort

Prerau et al., [1990] offer a variety of advice for designing maintainability into ESs including the recognition that items that are known to change regularly (e.g., items like VAT, interest rates, foreign exchange rates) should be located in external databases, not coded into the knowledge base. For example, ELSIE holds all its information on the costs of building materials, labour rates, and similar items in external databases. This significantly reduces the complexity of changing this knowledge, and therefore considerably reduces the time spent on evolutionary maintenance.

TOWARDS A METHODOLOGY FOR MAINTENANCE

Our case study has enabled us to begin formulating a methodology for maintaining ESs based on commercial experience. As in ED1 of the CCA [Watson et al., 1992] the stages of the methodology are represented by deliverables, not activities. Each deliverable is described below:

Record of desirable changes — This is the result of the ongoing activities of recording bugs, user feedback, and researching changes in domain knowledge while the ES is in use.

Prioritised list of changes — Assuming limited resources an organisation will have to decide the priority of changes to the system. In our study three lists of low, medium, and high priority were prepared. It is essential though that changes to the knowledge take priority over functional enhancements to areas like the interface. An ES must be judged by the quality of its knowledge, and not by its interface.

An understanding of the existing system — In our study the maintainer only spent four hours familiarising himself with the ES. He was however already very familiar with the domain, the system, and its code. Someone with less prior experience would be advises to gain an understanding of the following areas:

- the usage context of the ES,
- the original specification of the ES,
- the knowledge representation language and relevant software tools,
- the functionality of the ES,
- the user interface (including interfaces to other systems),
- sensitive parts of the knowledge base,
- obscure reasons why parts were programmed as they are (this will be assisted if records of why decisions were taken have been kept), and
- all the system documentation.

This familiarisation process could take several months.

Definitive version of system — This involves obtaining all the latest versions of the files that constitute the ES, and all the documentation that accompanies the ES. They must all be backed up, and version control procedures must be established. In our case study this work took almost one week.

Working system — In the CCA this is an ES that is complete and correct, i.e., all bugs in the knowledge base have been corrected and all the planned for modifications made to the knowledge base have been made and verified.

Usable system — This version of the ES takes into account enhancements to the system's functionality requested by the users, and the fixing of bugs in the user interface.

Saleable system — This version of the ES is complete with new installation routines, file conversion kits, and updated user documentation.

Embedded-in-use — The final deliverable is the new version of the ES embedded in use. The ongoing activities of recording bugs, user feedback, and changes to knowledge must now be re-established.

The sequence of the deliverables and the activities they require are outlined in the diagram below.

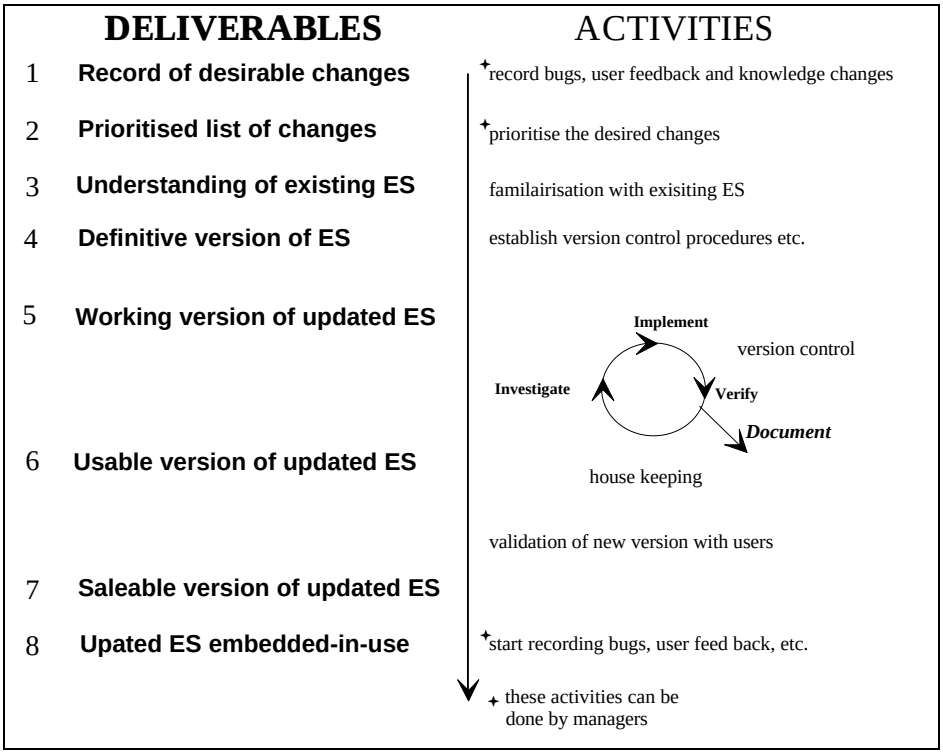

Figure 4. A Maintenance Methodology

CONCLUSION

Maintenance should be planned for if an ES is to be kept in regular beneficial use. This advice is supported by the findings of the recent DTI KBS survey [DTI, 1992]. Part of this planning must take place during system development (what we term ED1) and should seek to establish accurate but concise documentation of the system's requirements, its architecture, and the knowledge included in it. The maintenance of an ES involves the same

activities as with conventional software (i.e., adaptive, corrective, and perfective maintenance) with one vital addition. That is, knowledge changes over time, and this requires evolutionary maintenance.

Common sense and good programming practice are necessary to develop ESs for maintainability. Moreover, comprehensive records of design decisions should be established, including details of all the knowledge included in the ES [Neches et al., 1984]. This can be in the form of knowledge dictionaries [Recogzei, & Plantinga, 1987; Jansen & Compton, 1989], intermediate representations [Berry & Broadbent, 1986; Johnson, 1989; Watson et al., 1989] or other forms of documentation.

ES designers might also consider techniques such as rule induction [Al-Attar, 1991; Liu & White, 1991], automated maintenance of the knowledge base [Irani et al., 1990], and automated maintenance of the software itself [Pau & Kristinsson, 1989]. However, some of these techniques are still experimental and are unlikely to be practical within the context of SMEs for many years

New knowledge is added periodically to ESs throughout their operational life. The additions increase both size and complexity of systems. The efficiency of maintaining and updating ESs may be improved by using metrics that monitor the use and stability of knowledge. Statistics on the confidence in knowledge can help knowledge engineers evaluate how systems are used.

We believe that intermediate representations, particularly if supported by tools, can significantly improve the maintainability of ESs since it is the knowledge that needs to be maintained, and not the program code per se [Bench-Capon & Coenen, 1992b]. Maintaining the code is a consequence of maintaining the knowledge, not the reverse. This realisation is what differentiates the maintenance of ESs from other computer programs.

ACKNOWLEDGEMENTS

This research was funded under the IEATP Program, Project No.IED4/1/2062. We would also like to acknowledge our collaborators: The Royal Institution of Chartered Surveyors, Inference Europe Ltd., and Imaginor Systems, and our project user group. In particular I would like to thank Mark Forskitt of S$_f$K Technology Ltd, Lawrence Poynter of Inference Europe Ltd., and Frans Coenen of Liverpool University for providing valuable references and insights into ES maintenance.

REFERENCES

Al-Attar, A. (1991).
Rule induction from mythology to methodology. In: Research & Development in Expert Systems V111, (eds. Graham, I.M., & Milne, R.W.), pp.85-103. Cambridge University Press, Cambridge.

Alexander, J.H., Freiling, M.J., Shulman, S.J., Staley, J.L., Rehfuss, S., & Messick, S.L. (1986).
Knowledge Level Engineering: Ontological Analysis, in AAAI-86 vol.**2**: pp.963-968.

Bachant, J. & McDermott , J. (1984).
R1 Revisited: Four Years in the Trenches. The AI Magazine, **5**(iii).

Barr, A., & Feigenbaum, E.A. (Eds.) (1982).
The Handbook of Artificial Intelligence, Vols I & II. William Kaufmann, Inc. Los Altos, US.

Basden, A. (1989).
A Client Centred Methodology for Building Expert Systems, in People and Computers, V. Sutcliffe, A., & Macaulay, L. (Eds.), Cambridge University Press, Cambridge, UK.

Bench-Capon, T.J.M., & Coenen, F. (1991).
Practical Applications of KBS to Law: The Crucial Role of Maintenance. In, Legal Knowledge Based Systems, Aims for Research and Development, van Noortwijk, C., Schmidt, A.H.J., & Winkels, R.G.F. (Eds.), pp.5-17. Koninklijke Vermande, BV, Lelystadt, Netherlands.

Bench-Capon, T.J.M., & Coenen, F. (1992a).
The Maintenance of Legal Knowledge Based Systems, AI Review, **6**: pp.129-143.

Bench-Capon, T.J.M., & Coenen (1992b).
Isomporphism and Legal Knowlege Based Systems. J. of AI in Law. in print.

Bentley, P., Grisoni, M., Skingle, B., & Horsefield, M. (1992)
VORTEX II Final Report Part II: Knowledge Base Validation, Logica Cambridge Ltd. CAM:706.70062 - FR II D

Berry, D.C., & Broadbent, D.E. (1986).
Expert Systems and the Man-Machine Interface. Expert Systems, **3**(iv): pp.228-31.

Brandon, P., Basden, A., Hamilton, I., & Stockley, J. (1988)
Expert Systems: The Strategic Planning of Construction Projects. Royal Institution of Chartered Surveyors.

Bratley, P., Fremont, J., Mackaay, E., & Poulin, D. (1991).
Coping with Change. In, Proc. of 3rd. Int. Conference on AI and Law, Oxford 1991, ACM Press, pp.62-8.

Brooke, S., & Jackson, C., (1991). *Advances in Elicitation by Exception*. Proc. Ist SGKBS Int. Workshop on Knowledge-Based Systems Methodologies.

Butler, A., Chamberlin, G. (1987).
The Aries Club -- Experience of Expert Systems In Insurance & Invest-ment, in Research & Development in Expert Systems IV, pp.246-257, Moralee, D.S. (Ed.), Cambridge University Press, Cambridge, UK.

Caviedes, J.E. (1987).
Expert Systems Maintenance: Post-Implementation Support. IEEE Montech '87 Conf. - Compint' 87, pp.21-4.

Chee, C.W.J., & Power, M.A. (1990).
Expert Systems Maintainability. Proc. of the Annual Reliability & Maintainability Symposium, pp.415-18.

Coenen, F. (1992).
A Methodology for the Maintenance of Knowledge Based Systems. In, Proc. of EXPERSYS-92, Niku-Lari, A. (Ed.), pp.171-76.

Coenen, F., & Bench-Capon, T.J.M. (1992).
Building Knowledge Based Systems for Maintainability. In, Proc. of DEXA'92, Valencia, in print.

Diaper, D. (1987).
POMESS: a People Orientated Methodology for expert System Specification, in Proc. 1st. European Workshop on Knowledge Acquisition for Knowledge Based Systems. Addis, T., Boose, J., & Gains, B. (Eds.).

Diaper, D. (1989).
Knowledge Elicitation: Principles, Techniques & Applications. Ellis Horwood Ltd.

Department of Trade and Industry (1992).
Knowledge Based Systems Survey of UK Applications. Report performed by Touche Ross and commisioned by the DTI, February, 1992.

Edwards, A. (1987).
Knowledge Elicitation Warts and All: A Case Study.In, Proc. of the British Accounting Association Conference, Glasgow - April 1987.

Hart, A. (1988).
Expert Systems: An Introduction for Managers, Kogan Page.

Hicks, R.C., (1990).
A Composite Methodology for Low Maintenance Expert Systems Development. In, Proc. of 3rd. Annual Hawaii Int. Conf. on System Science, Vol. **3**: pp.293-302. IEEE Computer Society Press.

Irani, E.A., Matts, J.P., Hunter, D.W., Slagle, J.R., Kain, R.Y., & Long, J.M. (1990).
Automated Assistance for Maintenance of Medical Expert Systems: The POSH AI Project. Proc. 3rd. Annual IEEE Symposium on Computer-Based Medical Systems, pp.275-81.

Jansen, B., & Compton, P. (1989).
Data Dictionary Approach to the Maintenance of Expert Systems: the Knowledge Dictionary. Knowledge based Systems, **2**(i): pp.14-26.

Johnson, N. (1989).
Mediating Representations in Knowledge Elicitation, in Knowledge Elicitation: Principles, Techniques & Applications, pp.179-194, Daiper, D. (Ed.), Ellis Horwood Ltd., Chichester, UK.

Kalos, A. (1992).
Knowledge Engineering Methodology for Industrial Applications. In, Proc. of EXPERSYS-92, Niku-Lari, A. (Ed.), pp.593-98.

Killin, J., Morgan-Gray, L., & Porter, D. (1991).
Knowledge Engineering Within Software Engineering - Similarities & Differences. Proc. Ist SGES Int. Workshop on Knowledge-Based Systems Methodologies.

Liu, W.Z., & White, A.P. (1991).
A review of inductive learning. In: Research & Development in Expert Systems V111, (eds. Graham, I.M., & Milne, R.W.), pp.112-126. Cambridge University Press, Cambridge, UK.

Madeo, L.A., & Levary, R.R. (1990).
On Maintaining and Updating Expert Systems. Expert Systems, May 1990, **7**(ii): pp.121-122.

Moriss, L. (1989).
Maintenance-Oriented Design of Expert Systems. ISA Transactions, **28**(i): pp.23-6.

Motta, E., Rajan, T., & Eisenstadt, M. (1989).
A methodology and tool for knowledge acquisition in KEATS-2. In, Topics in Expert System Design: Methodologies and Tools, Guida, G., & Tasso, C. (Eds.), pp.297-322. North-Holland, New York, US.

Neches, R., Swartout, W.R., & Moore, J. (1984).
Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of their Development. Proc. of IEEE Workshop on Principles of Knowledge-Based Systems. pp.173-83.

Newell, A. (1982).
The Knowledge Level. AI **18**: pp.87-127.

Pau, L.F., & Kristinsson, J.B. (1989).
SOFTM: A Software Maintenance Expert System in Prolog. J. of Software Maintenance: Research & Practice, 2(ii): pp.87-111.

Prerau, D.S., Gunderson, A.S., Reinke, R.E., Adler, M.R. (1990).
Maintainability Techniques in Developing Large Expert Systems. IEEE EXPERT, June 1990, pp.71-9.

Regoczei, S. & Plantinga, E.P.O (1987).
Creating the domain of discourse: ontology and inventory. J. Man Machine Studies, **27**: pp.235-250.

Shortliffe, E.H., (1976).
Computer-based medical consultations: MYCIN. North Holland, New York, US.

Soloway, E., Bachant, J., & Jenson, K. (1987).
Assessing the Maintainability of XCON-in-RIME: Coping with the problems of a Very Large Rule-Base. In, Proc. of AAAI-87, **2**: pp.824-9.

Swanson, E.B., (1976).
The Dimensions of Maintenance. In, Proc of 2nd. Int. Conf. on Software Engineering, IEEE Computer Society Press, pp.492-7.

Watson, I.D., Shave, M.J., & Moralee, D.S. (1989).
A Knowledge Analysis Methodology Using an Intermediate Representation Based on Conceptual Graphs, in Proc. 9th. Int. Workshop on Expert Systems & their Applications, **1**: pp.183-198.

Watson, I.D., Basden, A., & Brandon, P. (1992).
The Client Centred Approach: Expert System Development. Expert Systems, **9**.

Watson, I.D. & Norman, M. (1992).
*ProTest: A Knowledge Analysis Tool for the Development of Expert Systems***.** In, Proc. 4th. Int. Conf. on Software Engineering and Knowledge Engineering, IEEE Computer Society Press.

Wielinga, B.J. & Breuker, J.A. (1986).
Models of Expertise, in Proc. of the European Conference on AI, pp.306-318.

Young, R.M. (1987).
The Role of Intermediate Representations in Knowledge Elicitation, in Research and Development in Expert Systems IV, pp. 287-288. Moralee, D.S. (Ed.). Cambridge University Press, Cambridge, UK.