# Industrial use of safety-related expert systems

**Dr. David Robertson**
Division of Informatics,
University of Edinburgh,
80 South Bridge,
Edinburgh

**Prof. John Fox**
Advanced Computation Laboratory,
Imperial Cancer Research Fund,
Lincoln's Inn Fields,
London

28th April 2000

# Contents

# 1 SUMMARY

Experts systems are being used in safety-related areas and their use appears to be increasing, yet there is little evidence of a safety culture within the knowledge engineering community. We explore why this is the case and suggest some areas in which practice could be improved.

Section 2 introduces the basic concepts fundamental to expert system design. It also explains some basic problems with the styles of knowledge representation and inference which commonly are used. These include: the tension between declarative and procedural knowledge representation; the ability to predict system function; and the ability to guarantee internal consistency of inferences. The knowledge engineering field has broadened into many sub-fields which, although related, employ different architectures for problem solving. We briefly summarise the most common of these: probabilistic networks, argumentation, fuzzy logics, qualitative and model based reasoning, planning, inductive machine learning, case based reasoning and neural networks. Finally in this section we highlight the fact that expert system design lifecycles are seldom sufficiently controlled from a safety point of view and relate this to a lifecycle model familiar to many safety-critical software engineers.

Section 3 surveys current expert system usage. Five application areas are described: regulatory advice, hazard analysis and avoidance, decision support, monitoring and diagnosis and post-accident analysis. Examples of systems in each of these areas are given and we speculate on likely future directions.

Section 4 describes the sorts of safety gains which may be obtained in each of the application areas surveyed in Section 3. Most of these benefits are obtained through the deployment in software of expertise which would otherwise be unavailable to the activity on which the expert system is targeted, for example because experts are scarce or because the expertise needs to be deployed more rapidly than humans can achieve.

Section 5 then highlights some of the potential hazards in expert system applications. These are classified in terms of five basic causes. The first is mismatch between the way in which it is convenient to describe the high-level specification of an expert system and the manner in which it is implemented. The second is the potential for musunderstandings about the meaning of symbols in formally represented knowledge. The third is the problem of predicting and staying inside safe envelopes of system operation. The fourth is the possibility that the precision suggested by some formal notations may be an illusion in terms of the real world systems represented. The fifth is whether it is possible to predict human responses to unanticipated expert system behaviours.

Section 6 considers the ways in which safety arguments might be made for expert systems. It classifies these according to whether they are achieved via proofs of properties, testing, the construction process or by appeal to the experience of designers. Major difficulties are evident in all four forms of safety argument. These are daunting but may be surmountable.

Section 7 looks broadly at some of the current trends in expert system research which impact on safety. The first concerns the design process, which is becoming more eclectic in its choice of methods, tools and languages. The second concerns architectures, for which there is now increased interest in designs allowing greater degrees of autonomy in

environments where information quality may be poor. The third is the embedding in hardware of systems which might hitherto have been implemented in software.

Section 8 identifies key centres of excellence in expert systems design. To our knowledge there is no significantly sized centre of excellence specialising in both expert systems and safety. This compares unfavourably to the picture in software engineering, where specialist centres exist and where safety is a theme in numerous institutions.

# 2 KEY TECHNICAL CONCEPTS

The term "expert system" was coined in the 1970's to reflect the aspiration of a (then) small group of researchers to replicate by computer the problem solving expertise of human specialists in narrow areas of application. As experience was gained in building these types of system it became accepted that design of successful expert systems is seldom as straightforward as replicating directly an expert's problem solving algorithm. Often the complexities of expert problem solving and the need to integrate the expert system with other system components and operators means that one replicates only part of an expert's problem solving abilities, perhaps integrating these with other computational techniques to provide a different way of addressing the problem. Because of this shift it is more common nowadays to find these sorts of systems described as "knowledge based systems" and those with training in constructing them referred to as "knowledge engineers". Nevertheless, we stick to the older term "expert systems" here because it is still commonly used by the safety community.

## 2.1 Basic Concepts

Central to most styles of expert system design is the idea that the knowledge which helps a system solve a problem can be represented separately from the mechanisms used to draw inferences from that knowledge in order to solve problems in some domain. This separation has two key engineering benefits: it enables designers to re-use similar inference mechanisms with different knowledge bases and it allows maintenance of knowledge bases without adaptation of inference mechanisms. To maximise these benefits it is desirable that knowledge is represented declaratively - that is, it can be read as a statement of fact independent of the machinery used to draw inferences from it. This does not mean that any inference mechanism can be used with any knowledge base. Each inference mechanism makes assumptions about the formal language used in knowledge representation so even if a knowledge base is declarative it must be expressed in the style appropriate to the appropriate inference mechanism in order to be used effectively. For this reason, knowledge representation and inference are more tightly connected in practice than we would expect them to be in theory.

### 2.1.1 Knowledge Representation

One of the earliest, and still commonly used, forms of knowledge representation is through production rules. These represent knowledge as `IF --> THEN` structures in which the `THEN`

part can be added to the working memory during inference if the `IF` part can be established from the current working memory. This sort of rule does not have a straightforward declarative interpretation because whether or not a rule applies depends on the contents of the working memory (and hence the state of the system) at a given point in time. Since each rule may alter the working memory, the order of application of rules matters and control information (such as rule precedences) is normally built into the knowledge base to guide the inference mechanism in its choice of rule. For example, we might have a system which monitors a process and is intended to infer an `alarm` outcome if there is a problem with the process and a `normal` outcome if there is no problem with the process. It does this by relating these outcomes to conditions (`condition1` and `condition2`) using the following three rules:

**Rule 1:** `condition1 --> feature1` with high precedence.

**Rule 2:** `feature1 --> normal` with high precedence.

**Rule 3:** `condition2 --> alarm` with low precedence.

Suppose `condition1` and `condition2` both hold, so they are in the working memory. Two rules can now fire (Rule 1 or Rule 3) but Rule 1 has the highest precedence so it is chosen and its conclusion, `feature1`, is added to the working memory. On the next cycle, all rules could potentially fire but Rule 1 has been applied so is discounted and Rule 2, having higher precedence than Rule 3 is chosen. We thus conclude the outcome `normal`, even though there were sufficient grounds for concluding the `alarm` outcome. In this example the problem is obvious and there is an easy fix but in more complex systems such errors may be difficult to detect and non-trivial to correct.

Representation schemes based on logic adhere more closely to the declarative interpretation of knowledge representation structures. In theory, this counters the sorts of problems we saw above when we mingle procedural and declarative knowledge in the same knowledge representation. In practice, we must sooner or later face procedural problems in performing inference (that is, we do have to have a procedure for choosing axioms in a logic when solving a particular problem) and the value of logic is in allowing a large part of the design, although not all, to be separate from our choice of procedure. The most common forms of logic for expert systems use logical connectives for implication, conjunction (and) and disjunction (or) to connect either constants representing statements about a problem (as in propositional logic) or structured expressions relating elements of a problem (as in predicate logic).

Expert systems often perform model based reasoning, where descriptions of a system are used to infer solutions to a problem, or case based reasoning, where examples of previous problem-solution pairs are used as paradigms for solving new problems. Object oriented design is natural for both forms of expert system and is particularly pronounced in frame based systems which often combine representational styles from production rules or logic with ideas of modularity and inheritance from object oriented design.

With any of the above representations we may need to represent uncertainty, in knowledge or in the inferences we make. Normally this involves labelling the elements of our knowledge base to indicate the form and level of uncertainty associated with them and then

propagating these measures of uncertainty during inference. Work on uncertain reasoning has been dominated by numerical methods but richer notations are emerging, partly in recognition of the difficulty in expressing some forms of uncertainty in a purely numerical way. The practical impact of these richer notations is currently small but it appears to be growing (an example appears in Section 3.3).

### 2.1.2 Inference

A multitude of inference mechanisms have been built but there is no consensus on how precisely these may be classified. The picture is further clouded by the fact that many systems use more than one style of inference (and perhaps also numerous styles of knowledge representation) to deal with different aspects of the application - such systems are often called "hybrids". Nevertheless, there are some broad distinctions which may be made from the point of view of safety. These are summarised in Figure 1.

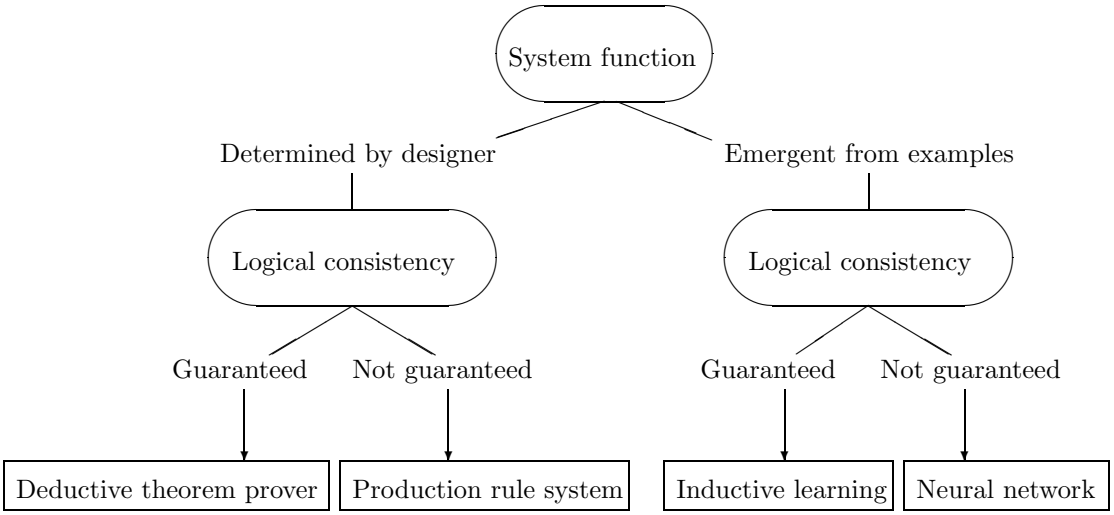


Figure 1: A high level classification of systems (terminating in examples)

At the highest level we divide expert systems according to the extent to which the original system designers determine the function of the system in use. If both the knowledge base and the inference mechanism are defined by the designers then it is normally possible to predict many of the behaviours of the resultant system and the safety argument (ignoring for the moment issues of maintenance) may be supported by relating these to the code. If our inference system is based on a formal logic (as, for example, in systems using deductive theorem proving) then it may be possible to present formal proofs of properties of the system. If not (as, for example, in production rule systems) then designers may still make safety arguments which relate to the structure of the code, in a similar style to conventional software engineers. By contrast, some expert systems are intended to determine their function according to the examples presented to them - this taking place either during training sessions and/or incrementally during operation. For these systems it is not possible to relate arguments about predicted behaviours directly to the code, although in the case of systems which use formal logic in an inductive style it may be possible to use formal proof to heighten confidence in reliability of function given assumptions about

the nature of examples. In the most extreme cases, such as traditional neural network based systems, the safety case must rely on statistical arguments based on testing and, in particular, the robustness of the system to new inputs for which it was not trained.

The above are all aspects of the safety case which concern the knowledge representation and inference components of an expert system. These are, indeed, distinguishing features of this sort of system but they are not the only components. For example it is common for expert systems to interact with human operators through a user interface, which raises questions about the interaction between the operator interface and the underlying inference mechanisms. Sometimes the need for enhanced interaction with operators leads to extensions in the inference mechanism. For instance, if operators require an explanation of the advice given by a system then this raises the technical issue of how to generate the necessary information as well as the human factors issue of how to present it. This broadens the safety case considerably and we are aware of very few safety cases at this breadth (one example appears in Section 3.2).

## 2.2 The Breadth of Modern Expert System Engineering

As Sections 2.1.1 and 2.1.2 explain, the bread and butter of expert system design involves symbolic knowledge representation (usually as rules or frames) and one or more inference mechanisms (such as a rule interpreter or a frame-based inheritance mechanism). A sophisticated expert system application, however, may require more specialised architectures. We summarise below the most common of these.

### 2.2.1 Probabilistic networks

In Section 2.1.1, we introduced the idea of propagating measures of confidence as we make inferences in an expert system. Many different methods exist for doing this but probabilistic network methods appear to be particularly effective in applications. The basic idea of a probabilistic network is as follows. We identify a number of propositions about which we have differing levels of confidence - these are the nodes in our network. We then identify pairs of propositions for which our confidence in one influences our confidence in the other - these are the links in our network. We select a method for propagating our confidence measures from one node to another, given the confidence associated locally with each node. This gives us our prepared network. We then apply the network to a problem by setting some of the nodes to the confidence levels associated with that problem (for instance, we may observe that some propositions are true in our problem and therefore set the corresponding nodes to the highest confidence level). Our selected confidence propagation method then propagates the results of this change in confidence through the network and we can examine the new confidence levels on whichever nodes interest us.

This method has been effective because, for problems which suit it, it provides a simple structure for knowledge representation and it allows flexibility in the ways in which confidence measures are propagated. Some of these have their origins in mathematical logic, particularly in Bayesian probability theory, so it is possible to make mathematical arguments about the correctness of their inferences. Others, such as Monte Carlo based networks, rely on statistical sampling from probability distributions derived from multiple

runs of the network so correctness arguments for those are empirically based.

### 2.2.2 Argumentation

Expert systems are normally applied in areas where conclusions drawn from inference are not certain. Therefore the steps we take during inference, even if they are performed using a system of logical proof, may not when interpreted in the real-world have the conclusive force of proof. Instead they are more appropriately viewed as arguments for particular courses of action. This has spurred experiments into the use of argument structures, rather than summative measures of confidence, to provide more sophisticated forms of control over the acceptability of inferences. An example is Imperial Cancer Research Fund's CAPSULE system which is a decision support system for the prescription of certain kinds of drugs. Typically, there may not be a single "correct" prescription but a number of different prescriptions, with arguments for and against each given patient symptoms. CAPSULE presents these arguments and ranks the candidate prescriptions in order (using a particular formal model of argumentation to do so). In tests with CAPSULE on doctors, they appear to prescribe more accurately and a little more quickly using this form of support and they often select a cheaper but equally effective form of prescription.

Argumentation systems appear to be most effective in circumstances where flexibility is required in weighing up evidence for alternative courses of action - for instance in choosing between alternative therapies in a medical protocol. These are often the conditions where traditional methods, such as numerical probabilities, may fail because probability values on their own do not give sufficient information to assess alternative strategies in terms which can be checked against standard practice by human experts. If well engineered, this approach may be helpful in avoiding the problem of false precision raised in Section 5.4. It is not, however, a panacea for this problem because the representations needed to infer judgements about uncertainty in argumentation systems (although expressed in a generic framework) usually require domain specific heuristics which may themselves be open to debate.

### 2.2.3 Fuzzy logics

Fuzzy logics are of particular relevance to safety engineers because they are commonly used in the control systems for electro-mechanical devices. From the expert systems point of view they provide yet another way of propagating uncertainty during inference. The main distinguishing feature of a fuzzy logic is the way in which it relates observations in the real world to measures of confidence within an expert system. This is normally done by using mathematical functions to translate the observed measurement to fuzzy truth values with associated probabilities - for example a temperature measurement of 25 Celsius might be translated into a truth value of "medium" with probability 0.6 and a truth value of "high" with probability 0.5.

This way of thinking about the relationship between observation and approximate reasoning seems to be effective in some engineering cultures (such as in control engineering). The key difficulty with fuzzy logic is that the "probabilities" associated with fuzzy values need not always behave as conventional probability theory would predict (notice in our

example the probabilities of being "medium" or "high" sum to more than 1). This does not necessarily create an engineering problem because an engineer can use combination functions which are well behaved under these circumstances (typically, the fuzzy value for a disjunction might be the maximum of the component values so for our example we would obtain 0.6 instead of 1.1). Nevertheless, it then becomes difficult to understand probabilistically the meaning of uncertainty measures in fuzzy systems.

### 2.2.4 Qualitative simulation and model based reasoning

The earliest expert systems were concerned with representing the problem solving procedures of experts but it was observed that beneath this surface behaviour there often was a model of the system about which experts were reasoning. For example, if we want to build an expert diagnostic system system for a class of electronic circuits we could either represent experts' diagnostic procedures directly or we could build computational models of the types of circuits and derive expert diagnostic behaviours from those. This type of model based reasoning has the advantage of relating inference more explicitly to the object of study, which can be an advantage if in-depth explanations of the reasons for conclusions are required. Sometimes it can also be simpler to engineer because in some applications the most natural way of expressing expertise is to relate it to a system model.

One of the most effective uses of model based reasoning is in qualitative reasoning when approximate models of a system (using qualitative rather than precise values for state variables) can be used to predict the large-scale behaviour of systems and these predictions can then be used to derive information which was not immediately obvious. For example, at Computer Science in Aberystwyth qualitative models have been used to automate some forms of fault detection for automotive electronic circuits. This has been applied to failure modes and effects analysis in Ford cars[1]. An advantage of this sort of qualitative analysis is that they normally have a finite set of model variables, each of which must take a value from a small set of possible values. Exhaustive exploration of the state space may therefore be possible. The downside of this is that by increasing the granularity we also lose precision so we need to be sure that important behaviours have not been lost in making the approximations.

### 2.2.5 Planning

Planning systems are concerned with the issue of how to decide on appropriate sequences of action for a given problem. What makes planning different from mainstream expert system engineering is the strong temporal element to planning tasks. Planning systems use temporal representations of differing levels of sophistication, from simple linear sequences to non-linear interval-based representations. They may also place different emphasis on mathematical logic - some systems having no direct connection to logic; others using forms of temporal logic to represent and reason about plans.

Although much of the early planning work was concerned with the efficiency of automated planning, many of the recent successes for planning systems have been in support roles

[1]Source: http://www.aber.ac.uk/ dcswww/Research/arg/fmeaprojects/flameproject.html

for human operators. That is, they assist rather than automate the formulation and enactment of plans.

### 2.2.6 Inductive machine learning

From the early years of expert systems research it has been recognised that expert knowledge often is difficult to acquire simply by observing or questioning experts. It is, however, sometimes possible to obtain problem solving knowledge through induction, by observing examples of behaviour and generalising from these. For instance, by studying a large number of examples relating primary faults to safety-related consequences we might be able to induce a small number of general rules which are consistent with the examples but also (because they are generalisations) allow conclusions for faults other than those we have already seen.

Effective inductive machine learning needs the right sort of application domain in which examples provide good characterisations of what induced problem solving rules should cover (so called positive examples) and what they should avoid (negative examples). A bugbear of automatic induction systems is sensitivity to this training set, with slightly different sets producing widely different generalisations. This problem is compounded by the fact that automatically derived generalised rules may not easily be understood by human experts so validation against standard practice can be a problem. Nevertheless, if one finds the right sort of problem then induction can generate useful knowledge bases, and there is evidence of the approach filtering into safety related areas. For instance the Turing Institute in Glasgow advertised their use of machine learning in process control operations in a nuclear power station[2].

### 2.2.7 Case based reasoning

Often human problem solving doesn't laboriously follow chains of reasoning from asssumptions through to conclusions but seems to work form recollections of past situations which are then re-interpreted to suggest solutions for similar problems. The idea of case based reasoning is to automate this style of reasoning by collecting a library of formal descriptions of past problems along with their solutions. New problems are then described in a similar formal language and a matching algorithm is used to compare features of the new problem to features of past problems in the library. The best matching past problem and its solution is then retrieved and automatically adapted to show its relevance to the new problem. If it is considered appropriate, the solution is then refined and the new problem-solution pair is stored in the case library where it may be re-used (hence case based reasoners are learning systems).

The effectiveness of a case based reasoner usually depends on its matching algorithm. This is often heuristic, being adapted to suit the type of problem being tackled and not always giving what is intuitively the best match. Matching algorithms vary widely in complexity, from simple keyword matching to complex systems of structural comparison and inference. The choice of best algorithm depends on the domain of application - abstract theories cannot decide this issue and only empirical testing can raise confidence that the matcher

[2]Source: http://www.turing.gla.ac.uk

is working acceptably well. We have been able to find no really convincing examples of case based reasoning in safety-related areas[3] but there are well known research prototypes, such as the CASEY system which does limited forms of cardiac diagnosis.

### 2.2.8 Neural networks

In traditional expert systems, expert knowledge is hand coded by knowledge engineers. In induction systems and case based reasoners (Sections 2.2.6 and 2.2.7), knowledge is automatically acquired so engineers do not transcribe it but they can interpret the elements of a decision in human terms because there is a symbolic system of knowledge encoding. In neural network systems, there is no symbolic system of knowledge encoding. A system of interconnected elements, normally with simple states of activation or deactivation given adjustable thresholds, is trained on a series of examples and "rewarded" or "punished" depending in whether its responses match those expected. In response to this it self-adjusts, normally by altering the activation thresholds of its nodes. It is possible for some types of neural network to reconstruct symbolic descriptions consistent with activation patterns but, in general, this is seldom done. This means that prediction of behaviour (and hence the safety analysis) of neural networks relies on statistical arguments related to the training sets and the structure of the network.

This difficulty in predicting the function of neural networks makes them particular objects of suspicion by those concerned with dependability. Nevertheless, this technology has proved surprisingly effective for tasks which resist symbolic representation, such as imprecise pattern matching or control in unpredictable environments.

## 2.3 The Design Lifecycle

There is no satisfactory large-scale software lifecycle model for expert system development from the safety point of view. Instead there are numerous small scale methods which apply to particular stages in design. Nevertheless, since expert systems are (mostly) software, and are frequently components of more traditional software projects, it is illuminating to relate these to a general-purpose software lifecycle model. For this, we have chosen the "V" model which is a basis for software safety standards such as IEC 61508. This is shown diagrammatically in figure 2. The basic idea is that design proceeds downwards through the left of the V (from description of initial requirements through architectural commitments to specification of system function and then to implementation) whilst analysis proceeds upwards through the right of the V (verifying that the application correctly implements the specification and architecture and validating that the requirements have appropriately been met).

We consider below each phase in the diagram:

**Requirements:** In the past most expert system projects restricted their requirements statements to something like "We wish to replicate the performance of expert X performing task Y". More recently, structured methods for describing standard

[3]This does not mean they don't exist (see comment in Section 3)

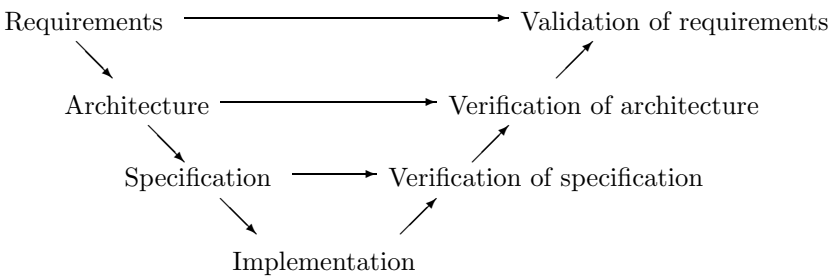


Figure 2: A general-purpose lifecycle model

types of problem solving have emerged - most notably the KADS method. These attempt to describe standard classes of problems which expert system builders are likely to encounter and to use these as a starting point for subsequent architectural choices. Such methods are frequently used in industry but raise two major safety issues. The first of these is the extent to which it is possible to link methods like KADS to methods for establishing safety requirements. The second issue is the way in which these safety requirements interact with architectural choices.

**Architecture:** Although hybrid systems are possible, the major architectural choice is normally between a symbolic expert system, in which knowledge is represented explicitly in some form and problem solving controlled by an algorithm which accesses that knowledge, and a connectionist (or "neural network") system in which neither the knowledge nor the problem solving method is explicit but the appropriate behaviour is "learned" by training on examples. This choice has a major impact on how designers may talk about safety. In the first case, it may be possible to prove formally (or at least argue in terms of the structure of the program) that the system will or will not give certain responses under particular circumstances. In the second case, the safety case is more likely to rely on statistical arguments based on testing and, in particular, the robustness of the system to new inputs for which it was not trained.

**Specification:** Many of the architectures used in expert systems design have been analysed, in theory, using formal methods but few of these formal accounts have found their way into industrial practice. Most expert system projects are driven by a cycle of prototyping and testing, where the nearest one gets to a formal requirements or system specification is an executable prototype in a high level language. There seldom is a fixed specification of the functional requirements the system should satisfy; nor is this identified formally during prototyping. Partly, this is because expert system developers are seldom pressurised to analyse their systems in a formal way (for example by demanding proofs of correctness in a high integrity system). However, it is probably still true that much of the development culture is driven by the desire to produce systems which surprise by their cleverness rather than convince us of their reliability (Section 6 gives a guide to the sorts of safety argument one might make for existing systems). A major safety issue is the extent to which existing prototyping languages have counterparts in formal languages which would allow us to discuss faulty as well as correct behaviour.

**Implementation:** Often expert system development takes place with high level prototyp-

ing languages and the deployment of the system is either within a restricted version of the original language or via automatic translation to some lower level language. This raises the safety issue of whether the restriction/translation preserves the behaviour originally specified, whilst introducing no new behaviours, and the related issue of whether the procedure from prototype to implementation is inspectable.

**Verification:** Most expert systems separate problem solving knowledge from inference mechanisms. This is often useful because it is common to change the knowledge base over the lifespan of the system and sometimes more than one problem solving method may be applied to the same knowledge base. This raises the question of whether verification must be performed every time the knowledge base is revised (which may be frequent). The answer to this may depend on the type of problem solving methods used because some will be more resistant than others to changes in the knowledge base. These questions are safety issues related to the more general issue of maintaining the integrity of knowledge based systems over time.

**Validation:** Most of the expert systems described in Section 3 support a human in some task. Therefore an appropriate validation question is not "Does the system replicate the performance of expert X performing task Y ?" but is instead "Does the system safely support operator X ?" This returns us to the first issue of the requirements phase: the need to relate expert system requirements methods to established methods of hazard and safety analysis.

# 3 USES OF EXPERT SYSTEMS IN SAFETY-RELATED AREAS

In the early days, expert systems were largely uniform both in terms of architecture (usually rule based) and type of application (usually diagnostic). They are now diverse on both counts. Sections 3.1 to 3.5 describe some of the types of application in which expert systems are now being applied. Each section gives an overview of the area of application; then supplies some typical examples of developed systems; and concludes with pointers to related UK research (Section 8 discusses UK and international expert systems research more broadly).

The examples given are a small fragment of those which probably exist, drawn from a variety of sources: Web searches; published accounts of applied research; and personal contacts of the authors. Where possible, we have included pointers to sources on the Web for further information on each example system. There are certainly many important examples which we have missed, since expert systems often are developed purely for in-house use by companies. These are normally treated as assets by those companies and information on them is hard to obtain without inside knowledge. For example, we know of a company which uses diagnostic expert systems for its aircraft engine maintenance support but does not advertise this because it sells the diagnostic service, not the technology.

## 3.1 Regulatory advice

Expert systems for regulations or safety guidelines normally adjunctly support existing textual documentation, with the expert system helping people to locate appropriate parts of documents and perhaps also advising on procedures for compliance but not supplanting the textual regulations. This is because it is almost impossible to convert an informal textual document (and even well structured regulations are informal) into a formal knowledge representation language without compromising some of the meaning of the original regulations. It is often practical, however, to represent in a formal way a part of the regulatory text and use this as a means of navigation when using the text for a prescribed purpose (such as checking conformance).

### 3.1.1 Examples

- The U.S. Department of Labor Occupational Safety & Health Administration provides a number of "Expert Advisors", most of which advise people about their responsibilities under particular guidelines or standards[4]. For instance, the Confined Spaces Advisor gives guidance on the Permit Required Confined Spaces Standard, helping to identify which areas are classified as confined spaces and assessing whether a permit is required.

- Dakota Decision Support Software market the Dakota Auditor which is a system for interactively checking compliance with regulations[5]. It contains a question asking system which drives the dialogue with an operator and, depending on the answers highlights appropriate segments of the textual regulations.

### 3.1.2 Future Directions

Work in this area tends to involve striking a compromise between the intuition and sophistication of interpretation inherent in natural forms of communication and the high levels of automation possible with entirely formal knowledge representations. One way of cutting through this problem is to begin formalisation earlier in the drafting of regulatory documents, using controlled forms of natural language which have a more direct translation to a formal knowledge representation scheme. An example of this is the ACE system[6] which uses a controlled natural language as a means of specifying a limited range of specifications described formally as logic programs. It has been shown that sometimes it is possible to reconstruct parts of manuals directly from formal specifications[7], giving an example of the more radical end of the formal spectrum in this area.

[4]Source: http://www.osha-slc.gov/dts/osta/oshasoft/

[5]Source: http://www.dakotasoft.com

[6]Described at http://www.ifi.unizh.ch/staff/fuchs/

[7]By Thimbleby and Ladkin, an example appears at http://www.cs.mdx.ac.uk/Harold/papers/log2manop.html

## 3.2 Hazard analysis and avoidance

Expert systems for hazard avoidance are intended to promote safety by advising on the extent of risks and the most effective forms of protection. They normally encode expert knowledge about how to provide that sort of advice in a narrow target domain. Unlike the document-based systems of Section 3.1, these systems normally are prescriptive so the accuracy of their recommendations is important. For this reason we see more attention being paid to design standards (knowledge engineering standards or safety standards) in systems which interact directly with people. Alternatively, their prescriptive advice may be mediated by a human operator with (one hopes) the experience to spot if the automatic recommendation is dubious.

### 3.2.1 Examples

- Adelard were funded by the UK Health and Safety Executive to produce an expert system (DUST-EXPERT) to give advice on protection methods for vessels that contain potentially explosive dusts. Adelard claim that a "full Safety Case is being produced to justify the development"[8], involving VDM specifications executed via the IFAD Toolbox; implemented in Prolog and supported by "a statistically significant quantity of testing".

- The Artificial Intelligence Applications Institute were funded by the UK Health and Safety Executive to produce an expert system (EASE) to guide risk assessment for workplace exposure to potentially hazardous new substances[9]. AIAI claim that the use of the KADS design method (see Section 6.3) helped standardise the design lifecycle and (it is implied implicitly in their description) raised confidence in the integrity of their design.

- 3M markets over 100 different respirator products designed for people working with any of 1700 different chemicals. Their Australian subsidiary now offers a free phone advisory service for people who want to know which type of respirator to use. The caller talks to a human operator who enters data into an expert system (built in collaboration with CSIRO, Australia) and relays the system's recommendation back to the caller. 3M aim to dispense with the operator and allow potential customers to interact directly with the system at points of sale[10].

### 3.2.2 Future Directions

The task for expert systems in hazard avoidance is normally to impart specialist expertise to non-specialists in hazard avoidance. This means that such systems must find ways to explain the principles familiar to specialists in a style which can be understood by non-specialists.

One way of doing this is by constructing systems which target particular engineering

[8]Source: http://www.adelard.co.uk
[9]Source: http://www.aiai.ed.ac.uk/~rhr/winease.html
[10]Source: http://www.iise.csiro.au

groups, as is the case in the SADI system[11] which gives safety advice for demolition engineers. These targeted systems often can deploy sophisticated combinations of media (SADI combines hypertext, photos and diagrams) in ways which are intended to have an intuitive appeal. An alternative form of delivery is to follow a general method which has proved effective across a variety of engineering groups. For example, hazard and operability (HAZOP) procedures are often used in different industrial sectors for hazard identification and it is possible to use knowledge engineering methods to support some of the more time consuming parts of this method[12].

Sometimes the conceptual gap between specialists and non-specialists can be particularly wide. An example is where specialists use mathematical models of hazardous events. The role of an expert system here is to act as an intermediary between the mathematical model and non-mathematicians by helping to parameterise the model and to interpret its results. An example is the SMARTFIRE system, produced by the University of Greenwich, which uses expert system techniques ... to simply simplify and automate the case set up and execution of simulations for a fire modelling system[13]. The claimed benefit is that this makes the mathematical models used in this form of modelling accessible to non-mathematicians such as fire fighters, architects or safety engineers, who don't have the training to parameterise and run the models themselves but who can learn from the simulations the models produce.

## 3.3 Decision support

Decision support places expert systems in contact with people who may have to take decisions in hazardous situations. In some cases development follows the traditional route of encoding expert knowledge in a stand-alone system, and the expert system supplies a second opinion for the people on the ground. In other cases the expert system may be tightly integrated with a larger system, including human operators, all of which must work in concert.

### 3.3.1 Examples

- CORETEC market an Expert System for Ice Management (ExSIM) which "helps oil rig personnel assess the threat represented by a particular iceberg and study the effect of alternative actions"[14]. It claims that the system "allows for sound decision-making in the absence of key personnel, since the expert's knowledge has been incorporated into the decision-making model".

- Atlantic Nuclear Services (a Nebraska based company specialising in reactor safety and licensing) lists in its product range "The Integrated Remote Monitoring and Diagnostic System" which "incorporates signal processing, statistical, physical and neural network models into an expert system for command and control"[15]

[11]Source: http://www.dis.port.ac.uk/∼bramerma/airg/kbs.htm
[12]Source: http://www-staff.lboro.ac.uk/∼cgpwhc/projects.htm
[13]Source: http://www-fseg.gre.ac.uk/
[14]Source: http://www.coretec.nf.net/services/emos/exsim.html
[15]Source: http://www.ansl.ca

- Infermed market a system called PROforma[16] which uses a formal knowledge representation language to describe clinical guidelines and protocols, also providing validation and consistency checking methods during composition, and can use these formal process descriptions to provide decision support during their enactment. It allows measures of uncertainty and risk to be propagated as decisions are taken, via an argumentation framework (see Section 2.2.2).

### 3.3.2 Future Directions

As the use of computers in general becomes more widespread in safety-related areas so the possibilities for involvement of expert systems in operators' decision making become greater. This raises issues of integration and of the lifecycle of the expert system as a component of the whole operational system.

Integration is becoming more of an issue because the style of design in significant areas of the expert system community is moving away from self-contained software packages to "agent" architectures where stand-alone expert systems are replaced by groups of problem solvers working together. This architecture may be more robust in environments where information is scarce and unreliable and allows a form of modular design which appears suited to distributed applications. However, this raises questions about how the behaviour of such systems may be predicted and incorporated into a safety case.

Related to the problem of integration is the issue of lifecycles for these systems. Some existing systems (such as PROforma) include part of the lifecycle within the formal method, in the sense that automated tool support is provided for composing guidelines (pre-operation) and for enacting them during operation. This is only a fragment of the full lifecycle (see Section 2.3) and making coherent safety arguments across the whole lifecycle remains an art rather than an engineering discipline.

## 3.4 Monitoring and diagnostic systems

In real-time monitoring, unsafe situations may be heralded by stereotypic patterns of sensor data. Expertise is required to judge whether these patterns are significant and sometimes it may be possible to represent this expertise formally. If so, it may then be possible to build the expert system into the monitoring loop. Taking this one step further, it may also be possible to automate some of the corrective actions needed to bring the system back under control. The demands of real-time engineering for these sorts of system limit the complexity of the techniques available and in this area especially the distinction between knowledge engineering and traditional systems engineering is blurred. Ironically, when we are very close to safety-critical situations we sometimes find expert system architectures, such as incremental learning, for which we may not be able to predict precisely the operation of the system before deployment (see Section 2.1.2). This happens because it is sometimes not possible to write down the decision procedure used by an expert in (say) the detection of significant patterns from a particular kind of sensor. These may only be derivable through experience and, in suitable types of application, expert systems based on induction may give adequate performance. What counts as adequate depends

[16]Source: http://acl.icnet.uk

on the context - in some applications any warning of potential problems may be better than no warning at all.

### 3.4.1 Examples

- The Argonne National Laboratory has produced a Multivariate State Estimation Technique (MSET) Expert System which gives early warning of faults on sensors, equipment and plant processors[17]. They claim their system "alerts plant personnel well in advance of warnings provided by conventional monitoring systems".

- The Intelligent Anaesthesia Monitor project at the Technical University of Delft has built a system for monitoring patient signals during surgery and can generate alarms when critical patterns appear in the signal data. They state that the incremental learning algorithm at the heart of their system is "now used in practice" although it is not stated whether their system has been used in real operating theatres.

- Expert systems have been used in prototypes for in-flight aircraft diagnostics but it is not clear whether this will extend to routine use. A NASA press release[18] of 10 years ago reports: "In the first flight in a joint NASA/USAF program that promises self-repairing flight controls and lower maintenance costs in future aircraft, computers aboard the NASA Ames-Dryden F-15 Flight Research Aircraft were able to correctly identify and isolate in flight a simulated failure in the flight control system. Flight control system failures can and do occur during flight. When this happens, costly ground maintenance diagnostic tests are conducted to try to identify the failure so that appropriate corrective actions may be taken. In many cases, the failure cannot be identified during ground tests because the actual flight conditions are not duplicated. With the new expert system technology, failures can be identified and isolated before landing and be fixed immediately."

### 3.4.2 Future Directions

The use of expert systems for monitoring and diagnosis has a long tradition, to the extent that some small companies specialise in this area of application[19] and many large companies have units with expertise in this area. Monitoring technology is becoming more diverse as applications of technologies such as neural networks becomes better understood. With this comes the need to integrate these different forms of inference system (some of which do not allow behaviours during inference to be related directly to system code) with other more traditional components of diagnostic systems.

A related issue is the extent to which the expert system elements of monitoring are allowed to interact with the control system. This is technically possible in real time for some systems but the resulting system complexity makes safety cases difficult to argue, except through defence in depth architectures which would contain control errors resulting from the monitoring system or by arguing that the additional sophistication would make the safety problem no worse.

[17] Source: http://www.itd.anl.gov/software/mset.html
[18] Source: RISKS digest Volume 8 Issue 67
[19] An example in the UK is Intelligent Applications, http://www.intapp.co.uk

## 3.5 Post-accident analysis and corporate knowledge

After an accident happens it is important that those concerned learn from the event. This requires that the right sorts of information are gathered at accident sites and that there is a distribution mechanism for information about the consequences. A major problem is volume. In some sectors (for example road transport), there are large numbers of accidents which normally are investigated by non-specialists in accident reporting. In other sectors (such as the aeronautics industry), there are large numbers of designers working on many different components over long time scales so the lessons from failure of earlier components can be lost. Expert systems may be used to reduce the reporting problem by automating standard accident investigation procedures. They may be used to tackle the corporate memory problem by providing assistance in accessing databases of product designs and associated failure or accident records.

### 3.5.1 Examples

- The US Federal Highway Administration is funding Veridian Engineering to build into their hand-held computer crash data collection system an expert system to "improve the quality and quantity of crash data, without overburdening the officer collecting the data"[20]. The aim appears to be to make the dialogue with the data collection officer more directed by incorporating expert guidance on the form of questioning relevant to features of an accident.

- Aerospatiale Missiles, in collaboration with the Norwegian Institutt for Energiteknikk, launched the ACCES project to tackle the problem of loss of key elements of design knowledge between projects[21]. It uses a combination of standard database and video technology with expert system techniques to assist designers in accessing the most relevant knowledge for a particular problem.

### 3.5.2 Future Directions

There has recently been major industrial investment in knowledge management for various forms of business-critical information. Engineering of knowledge management systems is normally an eclectic task, requiring expertise in knowledge engineering, databases, internet system design, and various other branches of information technology. Since post-accident analysis is less closely connected to safety-critical engineering there are no strong safety constraints on the forms of support which can be provided. Hence, engineering is limited more by commercial viability than by safety requirements.

The internet, and related standards and technologies for information interchange, makes large scale knowledge acquisition and distribution possible. It is technically feasible in some domains, for example, to collect accident data via expert systems on handheld computers; transmit this to central knowledge servers which combine accident information; then automatically generate some forms of accident reports which can be viewed on the

[20]Source: http://www.calspan.com/its.html
[21]Source: http://www.parallaxgraphics.com/solutions/aeros.html

internet. A major issue for the safety community, however, is whether this sort of technological advance works in practice to promote safer design or greater awareness of hazards.

# 4 POTENTIAL SAFETY GAINS

Section 3 describes application domains to which expert systems have been applied. For each of these we now summarise the benefits which might accrue.

## 4.1 Regulatory advice

Currently the main benefit in using expert systems for regulatory advice is to provide a more active form of advice giving than was possible with paper based regulations. This can range from simple assistance in document navigation to conformance checking based on formal descriptions of features of a client's problem. If engineered well, the benefit for those consulting regulations is that this should be easier and more thorough because irrelevant parts of the regulations may be automatically avoided and relevant details which might otherwise have been overlooked can be made more prominent.

An additional benefit, which often is claimed for expert system design in general but is more difficult to assess, is that producing versions of parts of regulations in a formal knowledge representation language forces authors of the regulations to think anew about the logical structure of those regulations. This may give them insights into the way regulations are described and applied.

## 4.2 Hazard analysis and avoidance

A key benefit of expert systems in hazard analysis and avoidance is delivery of domain-specific expertise to people who are not specialists and who therefore might not avoid those hazards without interactive guidance. Separation of knowledge base from inference mechanism allows knowledge about the hazard and its consequences to be expressed independently form the dialogue used in advice giving. This allows knowledge engineers to separate the task of mapping expert knowledge on hazards from the task of producing appropriate dialogue systems for target groups of non-specialists.

## 4.3 Decision support

Although expert systems may also have an educational role in decision support they currently seem to place greater emphasis on rapid and timely support of skilled or semi-skilled operators. Their benefit is therefore as an extension to existing skills, enabling knowledge which may not hitherto have been available directly to operators to be brought to bear on a problem. One reason why one might expect benefits from expert systems in this area is that careful mulling over of the pros and cons of particular decisions can be done before deployment of the system, and expressed in a knowledge base, then rapidly reconstructed once the system is deployed. In some circumstances this may enable operators to bring

into their decision making factors which would otherwise have taken them too long to assess.

### 4.4 Monitoring and diagnostic systems

The problem addressed by expert systems in monitoring is that many safety-critical systems produce large volumes of data in real time. Human specialists, given enough time, might be able to analyse these for features indicative of faults or other safety related features but there may not be enough time for them to do this when the system is running and, in any case, it may be impractical to expect a human to perform this sort of monitoring continuously. In such cases, it may be of benefit for part of the expertise used in interpretation of data to be represented formally and applied in real time as a monitoring or diagnostic system. Assuming that some niche can be found for this system in support of an operator then it may not need to have predictive abilities as good as a human expert to be of benefit - a small average increase in advance warning of aberrant behaviours may justify the investment in monitoring.

### 4.5 Post-accident analysis and corporate knowledge

One of the enemies of rigorous design is monotony. It is boring for engineers to produce documentation for their designs in various formats suited to other engineers. It is tedious to document failures and accidents in standard formats which are only partially relevant to particular engineering groups. It is time consuming to wade through voluminous documentation on old designs and their outcomes when trying to build a new one. This makes it unlikely that useful safety-related knowledge will be retained in an organisation and doubtful that it would be re-used by engineers because the overhead in doing this is too great. A benefit of expert systems is to reduce the overhead of acquiring safety knowledge by directing the acquisition of this information; then using the fact that it is formally represented to reduce the overhead of searching for relevant past cases which could be relevant to new designs.

## 5 POTENTIAL HAZARDS

Section 4 summarises some of the safety gains which may be obtained by using expert systems. We now consider the hazards which may ensue. Some of these are confined to the engineering methods used internally by expert system designers. Others raise issues of standardisation within application domains and of the match between expert systems and the wider systems in which they operate.

### 5.1 Mismatch between specification and implementation

In software design, it is normal for specification of a system, and associated proofs of safety properties, to be done in a language which is different from the one in which the system is implemented. We hope that mismatches between specification and implementation will

be caught during verification of the specification (see Section 2.3) but this is unlikely to provide a comprehensive guarantee of equivalence.

One of the ironies of expert system design is that, although a large amount of theoretical work has been done on languages which are relevent to their specification, formal specification seldom occurs in practice. Instead, design is normally done through prototyping (perhaps in a high-level language) and then implementing in either a compiled version of the prototype or by rewriting the system in a different language. This means that, in the first place, it may be difficult to prove properties of the prototype system and, in the second place, the link from prototype to implementation may not easily be inspected.

## 5.2 Ontological discrepancies

Knowledge bases are normally domain-specific which means that engineers must choose a vocabulary of terms which they believe describe meaningful concepts in the domain; then assemble these into structures which can be used as a basis for inference. The abstract formal theory used in verification of knowledge bases allows us to test whether the structures knowledge engineers invent are internally coherent with respect to a logical system. For example, if in some rule based system we have given a rule which says "temperature $> 80$ implies safety-hazard" and also a rule which says "temperature $< 100$ implies not(safety-hazard)" then a verification system might be able to detect a potential conflict between these rules in the event that $80 <$ temperature $> 100$. However, if our two rules say "temperature $> 80$ implies uncomfortable" and also a rule which says "temperature $< 100$ implies maybe-tolerable" then no automatic verifier will detect a potential conflict without more information, such as "not(uncomfortable and maybe-tolerable)". Although in our small example it may be obvious that we need to do this, in practice it is difficult to be precise in our use of language to the level required for extensive formal verification (for example, it is a matter of debate whether "uncomfortable" and "maybe-tolerable" should be mutually exclusive or should overlap in some way).

## 5.3 Straying beyond safe envelopes of operation

Expert systems are often deployed in application areas where the correct problem solving behaviours are not precisely known. This means that they must derive the best solutions they can with partial, heuristic information and with no clear definition of what a perfect solution might be. This makes it all too easy to construct systems which may propagate information which is unsafe because a heuristic which worked well in typical kinds of problems has been applied in conditions not envisaged by the designers.

An example of this is in systems using rule bases derived by induction (see Section 2.2.6). These knowledge bases are produced by generalising from examples. As well as giving answers consistent with these training examples we want the induced rules to be able to answer as many questions as possible which were not a consequence of the original training set. This is normally done by making the conditions of our rules less restrictive so they apply in more circumstances. The safety problem is that we do not normally have a specification of all the circumstances which we should exclude when attempting to generalise in this way. This means that our generalisation may inadvertently include some

of these. If it does then the answers we derive may be far different from those we might intuitively have expected. As we explained in Section 2.2.6, induction tends to be highly sensitive to the choice of training examples so information on a new example can radically alter the way in which we generalise.

We have used rule induction as an example of the hazards of generalisation but similar problems may appear in hand-coded knowledge bases. Although the most obvious source of these sorts of problems is in the knowledge base, some inference mechanisms may exacerbate the problem if misused. For instance in Section 2.2.3 we explained how fuzzy logics allowed overlapping fuzzy truth values. A skilled fuzzy logician will ensure that this overlap does not create anomalies during inference (such as probabilities of fuzzy set membership exceeding 1) but inadequately trained engineers may not, and the problem may not be detectable simply by inspecting each fuzzy rule since each may, independently, be consistent. The same sorts of problems await the unwary in other forms of uncertain reasoning system.

## 5.4 False precision

Some inference methods yield conclusions with high degrees of mathematical precision. It is tempting to equate this with actual precision in the real world but this may not always be the case. Sometimes the integrity of the initial information on which conclusions were based may be in question. At other times the inference mechanism itself may appear more precise than it actually is. We give an example of each below.

Suppose that our inference system uses a Bayesian style of uncertainty measurement which assigns each basic proposition a probability between 0 and 1. The normal mathematics of probability may be used to combine probabilities during inference and assign a probability to each conclusion of the system in a way which can be proved to be consistent with Bayesian theory. This gives a precise and (in terms of Bayesian theory) reliable measure of uncertainty but only if we have satisfied a number of assumptions about the problem we are modelling. In the first place, it must make sense to assign probabilities to our basic propositions. This is not always true because some forms of imprecision are not easily expressed as probabilities, although it may be tempting to pretend they are. Even if the imprecision in our problem is naturally expressed in probabilistic terms it is not always the case that simple probabilistic inference systems capture relevant interactions between propositions - a classic example being the assumption of independence between the propositions when in fact causal relationships have been conveniently ignored.

The example above explains why care is necessary in accepting high precision answers from expert systems with a high level of internal consistency supported by mathematical proof. Many expert systems, however, give seemingly precise measures of uncertainty as a consequence of more procedural (and hence less easily verifiable) methods. An example is a production rule system in which confidence factors are assigned to basic propositions and a rule firing mechanism propagates these confidence factors when inferring answers. Although the precision of the numbers produced by this system may look similar to those of the previous example there are additional problems. The confidence factor assignments tend to be assigned based on intuition rather than being justified in terms of statistical probabilities. This is not just sloppy engineering - it is often the case that a confidence

factor must be assigned without statistical analysis because that analysis may not be possible in the application domain. Even if the confidence factors are all coherent (a property which may be difficult to guarantee in practice) the inference mechanism in a production rule system may sometimes give different confidence factors for answers depending on the sequence of rules applied in deriving the answer, and engineers may fine-tune the choice of rules so that an expert system chooses just those ones which lead to the outcomes they feel are right for that domain. This means that a seemingly high precision result may be one of a number of possible results which may differ widely in magnitude, with the choice of which to output being made by procedural controls over rule choice during each run of the system.

### 5.5 Human responses to unpredicted behaviour

When used in support of human operators, the role of an expert system is normally to supply additional expertise which would not otherwise have been available to those operators. It follows that the advice given by the expert system may occasionally be surprising to an operator. There is then an issue of credulousness: should the operator take on trust what the system is saying or should further checking be done before acting? This issue has been a topic of debate within the safety community for some time - for instance when it was being suggested that expert systems might be used in some roles in battle command[22] - but there is no easy technical answer. Some architectural features of the expert system may help - for instance if it provides an explanation facility which is effective in justifying recommendations in terms of the components of the decision made by the system. Ultimately, however, the safety of this aspect of expert system operation depends on a range of factors specific to the application, not least of which being the mindset of the operators themselves.

## 6 COMMON SAFETY ARGUMENTS

Although safety arguments for expert systems could potentially involve many different forms of evidence, few companies make safety case production a focus of their activity. One reason for this is that production of convincing safety cases for these sorts of system is difficult and poorly understood. In Sections 6.1 to 6.4 we consider some of the problems related to commonly expected forms of safety evidence: proof; testing; process and experience.

### 6.1 Proof of safety properties

Since some expert systems use knowledge representation languages based on logic one might expect arguments for their safety to use formal proofs of properties of the system, such as that particular conclusions would always be reached under given circumstances or that particular conclusions would never be reached under given circumstances. In practice,

[22]Source: RISKS digest volume 10 Issue 37

proof is seldom used in this style by expert system designers. Several reasons may account for this:

- Knowledge engineers are seldom trained in theorem proving to the level needed for such proofs.
- It is not always clear what sorts of properties we would wish to prove of the system because, assuming it is a heuristic system, the proofs in which we would be most interested relate to conditions of use which are not well covered by the heuristics we built into the system. For example, in a rule-based advisory system we might encode a rule which suggests a particular action if some combination of easily verified conditions are observed. This may express a heuristic which is highly accurate in almost all circumstances but, because it is a heuristic, there may be some circumstances (involving more complex conditions than those required for the basic heuristic) where the heuristic cannot be trusted. Standard expert system design methods seldom involve modelling of these areas of faulty behaviour so the safe envelope of operation of the system may not be known with precision.
- If we prove properties of the knowledge base alone then the proofs we obtain for it may not be correct when it is used with an inference mechanism we did not anticipate. For instance if our knowledge base is expressed in a first order logic and we perform proofs using a deductive theorem prover (the most obvious choice) the properties we prove may not be preserved if we then use the knowledge base with an abductive inference mechanism - which is allowed to hypothesise plausible information which, when added, to the knowledge base, would account for an observation.
- Proving properties of the combined knowledge base and inference mechanism may be much more difficult because inference mechanisms normally are more complex mathematical objects than knowledge bases (for example inference mechanisms almost always contain recursion whereas knowledge bases may not). The best we may be able to do is to guarantee that those inferences which are obtained are correct with respect to a particular style of inference (normally classical or constructive deduction).
- Often an expert system must connect to other systems, for instance a window based display system if it is to interact with human operators. This means we may want our properties to include aspects of that larger system, not just the knowledge base and inference method. For instance we might want to prove for a monitoring system that under a particular combination of sensor readings the operator would be shown a red light within a given period of time. This expands the scope of the proof considerably, requiring specification of interface and timing aspects of the behaviour of the system.

## 6.2 Testing

Part of the testing of expert systems is similar to that for more conventional forms of software: the system is run on a suite of test examples and the outcomes compared to those expected. At this level, the normal problems of testing manifest themselves in new forms:

- The choice of examples is important because it is easy to build systems which over-generalise and testing with counter-examples is one way of detecting this.

- The corrective action necessary to avoid an error revealed during testing may be difficult to perform. In the first place it may be hard to decide where the source of the error occurred, as is normal in software. Even if we identify the source of the error there may be difficulties in correcting it. If the error is in a set of rules derived by inductive learning then we could include a counter-example and re-generate the rule set but this rule set may be very different from the one we tested (see Section 2.2.6) and some of the earlier tests might need to be done again. If the error is in a set of rules derived from consultation with human experts then even though we know in which formal rule the error appears it may be that the human experts prefer to keep that rule and revise others to fix the problem in line with their intuitions, so in this case the correction may not be entirely in the hands of the engineers.

## 6.3 Arguments based on the construction process

As we said in Section 2.3 there is no tried and tested safety lifecycle model for expert system design. In fact few lifecycle process models of any kind have proved successful in this area. One reason for this may be that the style of design has traditionally been anchored by the design of two artifacts - the knowledge base and inference mechanism - and other design activities have tended to revolve around the construction of these using standard formal languages. Nevertheless, there are some methods which cover parts of the lifecycle and have been used quite widely, although we are still a long way from industrial standards. The most commonly used method of this sort is KADS.

A KADS analysis involves four layers of abstraction. At the lowest level, the domain layer contains concepts and relations specific to the domain of application. Above this, the inference layer gives a declarative description of the problem solving processes used. On top of that, the task layer provides a procedural interpretation of the inference layer. Finally, the strategy layer connects together the problem solving tasks. Central to this arrangement is a library of interpretation models, which provide configurations of inference functions and knowledge roles appropriate to particular types of task. Once an interpretation model is identified, it is then adapted and instantiated in order create the inference layer of the KADS model. To assist users in classifying interpretation models by task, a hierarchy of tasks normally is provided. Reinterpreted in terms of the "V" model of Section 2.3, essentially what KADS does is take designers down part of the left arm of the "V" of Figure 2, from a partial requirements description through to a partial specification. Although it appears to be helpful in structuring and documenting designs, particularly at the architectural level, it (and other methods like it) remain problematic as far as safety is concerned:

- The links between different levels of abstraction are often informal. For example, having chosen an interpretation model to shape the architecture used in a system one is allowed to alter that model in ad-hoc ways so it is easy to end up with architectural models which do not closely resemble those from which they originated.

- There is only limited support for realising specifications expressed as KADS as im-

plemented systems in the sorts of languages and with the degree of reliability which would be expected in safety-critical applications. At this level of design KADS (like other methods of this type) currently is oriented toward executable prototyping rather than implementation.

- The horizontal links on the "V" model of Section 2.3 from design to validation and verification activities are not well supported in KADS, which is essentially neutral to the forms of validation and verification which might accompany it. There is scope to extend KADS (or a similar method) with specific methods for verification and validation. The best point of contact for these might be the interpretation models, assuming that there might be a correlation between the different forms of task and the forms of verification and validation that would be appropriate.

Despite these shortcomings, KADS does enforce a style of structured documentation to accompany design and this, in itself, heightens confidence that at least a significant portion of the design process would be inspectable.

Efforts are beginning to be made to connect some forms of design lifecycle to safety management. An example is the PROforma system (which we met in Section 3.3) which is based on a particular inference model. Associated with this inference model are various stages in its design and maintenance lifecycle. To each stage is attached forms of safety analysis and management which are appropriate to that stage and that inference model.

### 6.4 Arguments based on experience of engineers

Few companies have a substantial, long-term track record for building expert systems in safety-critical domains. However, limited forms of appeal to experience are still possible:

- A few companies have experience in both safety-related systems and in expert systems. These are able to produce more tightly focussed systems which relate more directly to their customer base.
- Some companies have a substantial track record of development of more general expert system software and emphasise the transfer of these generic skills to target safety-related domains.
- Some companies are more familiar with industrial methods in expert system design (such as KADS) and can emphasise this sort of process control experience, perhaps along with their more general design expertise.

## 7 TRENDS IN EXPERT SYSTEMS RESEARCH AND DEVELOPMENT

Activity in expert systems continues to expand, although few modern practitioners would think of their work purely as expert system design. Modern design of these sorts of systems has become more eclectic, in terms of the languages and methods used to support

design and in terms of the engineering cultures brought to bear on the problem. The drive for more adaptable, distributed systems is also stimulating work in systems which are less fragile when forced to make autonomous decisions in difficult environments. At an implementation level, the balance between software and hardware implementation is shifting as it becomes possible to embed more of the function of an expert system within hardware. We discuss these changes below.

## 7.1 Eclecticism in design

Before the surge of interest in expert systems research in the 1980's construction of expert systems often had to be done using programming languages which weren't particularly suited to that task. For engineers this meant ground-up programming of standard inference mechanisms which was time consuming. In reaction to this difficulty there was much academic and industrial activity in producing special purpose expert system design tools - in particular "shells", which were versions of particular expert systems stripped of their domain-specific features and adapted for re-use, and "toolkits", which are collections of standard expert system components offered in an environment which assists in their assembly.

Although popular for a time, few of these systems are in use today. One reason for this is that conventional programming environments are now faster and easier to use, making it less attractive for normal programmers to invest large amounts of time and money in shifting to a specialist programming environment. However, a more fundamental reason is that the knowledge base and inference mechanisms inside a modern expert system normally are just components of a much larger package which may include user interface and database components and may be ported to numerous platforms. Thus there is a substantial element of traditional software engineering in expert system development. More lightweight, portable development environments are better adapted to these conditions.

## 7.2 More autonomy in loosely constrained environments

The internet has brought strong demand for systems which are capable of operating in environments which may not be trustworthy and which may not yield all the information needed to take courses of action with confidence. The emphasis for knowledge engineers is shifting from constructing the best self-contained problem solvers possible, assuming high quality information about a problem, to constructing the most resilient problem solvers, assuming intermittent low quality information. The current label for this area is "agent" systems. Currently, the most thriving application areas for agents are not safety related - for example agents which roam the Web in search of certain types of information or agents which coordinate Web services such as auctions. However, significant efforts are now being made to design agent systems with the ability to reason about the safety of their actions with the aim of deploying them in safety related areas such as medicine[23].

Work on agent systems need not involve radically new notations but it does concentrate design effort on areas of the system which hitherto might have been taken for granted.

[23] A book on this topic by Fox,J. and Das,S., *Safe and Sound: Artificial Intelligence in Hazardous Situations* will soon be published by MIT Press

Consistency checking becomes important because it is assumed likely that either the state of the environment will change or assumptions made internally by the reasoner with minimal support will turn out to be false. Mechanisms for adaptation to these changes then become necessary, involving belief revision and perhaps re-planning of actions. Collaboration may be important if the information available to the agent is insufficient for it to solve a problem on its own. This requires protocols for knowledge transfer and perhaps also for negotiation and task allocation. The set of agents available to solve a problem may not be fixed so mechanisms for advertisement and brokering of capabilities of agents may need to be provided.

As far as safety is concerned, many of these new themes are interesting because they help to focus attention on a part of an expert system which is of concern to safety engineers: its interface to other systems. On the other hand, we know little about how to begin making safety arguments for large collections of semi-autonomous reasoning systems, which are the goal of large scale agent research.

### 7.3 Embedding in hardware

It is becoming possible to shift more of the functions of some kinds of expert system into hardware. The most common example is the use of fuzzy logic in control hardware for electro-mechanical devices such as washing machines or lifts. Much of the recent industrial momentum for applications of fuzzy logic in hardware has come from the development of cheap, mass-produced fuzzy logic chips. As customised chip design becomes easier we can expect other, and more sophisticated, forms of inference to be built into hardware rather than having to run as software. This may allow some forms of safety analysis from hardware design to translate to this form of embedded system, although there remain deep issues of consistency between the different forms of specification to which knowledge engineers and hardware designers are accustomed. On the other hand, the incorporation of systems which are designed to tolerate uncertainty and make approximate decisions raises many safety issues which are not resolved by changing the medium in which they are implemented (see Section 5).

Given the variety of methods for performing uncertain inference it is, perhaps, surprising that fuzzy logic is so prominent in applications which are directly in the control loop for hardware systems. One reason for this may be historical: part of the origin of fuzzy logic was in research groups familiar with control theory so culturally the translation of the methods to hardware control may have been easier. Another technical reason may be that many styles of dealing with uncertain inference in software systems are related to first-order logic rather than being propositional - the important difference being that first-order systems allow statements to contain variables where propositional systems do not. Translating from first-order systems directly to hardware can be difficult because the hardware must be able to represent variable binding (the objects to which each variable is associated during inference). If one wants to guarantee useful safety properties of the hardware system - such as response in finite, predictable time - then the translation requires some restriction of the class of first-order expressions which can be used. Ways of doing this have been explored but not extensively.

# 8 KEY CENTRES OF EXCELLENCE

Expertise in safety-related expert systems tends to be highly localised and there have been few attempts to understand in depth how the methods and standards recognised by the safety community in general (and the software safety community in particular) interact with expert system design methods and engineering styles. One recent European initiative in this area was the SAFE-KBS project[24], funded by the EU Esprit 4 programme, involving a consortium of European industrial sites. This was intended to develop "engineering methodology, certification guidelines and procedures for the development of safety-critical KBS" but the results of this project have yet to make an impact on the expert system community in general. One reason for this is that it takes time for the results of individual projects to propagate to the wider community. It is, however, also the case that the adoption of safety practices is extra effort for expert system developers and there is little incentive, either through peer pressure or regulation, for them to make that effort.

To our knowledge there are no centres of excellence which specialise in expert systems and safety. Therefore we have listed centres of excellence for knowledge engineering where the topics being studied are closely related to the design features described earlier in this report. Our lists are therefore subjective because they depend on what we consider to be close relevance. We have also favoured centres with long-standing, broad range of expertise in appropriate topics rather than targeting small research groups in otherwise irrelevant centres. This is because large centres of expertise tend to last longer.

[24]Source: http://www2.computas.com/research/safe_kbs.htm

Table 1: Centres of excellence in the USA

| Place | URL | Themes |
|---|---|---|
| Computer Science, Carnegie Mellon University | cs.cmu.edu | machine learning, agent systems |
| Artificial Intelligence Lab, Massachusetts Institute of Technology | ai.mit.edu | knowledge representation, AI in medicine |
| Computer Science, Rutgers University | cs.rutgers.edu | machine learning, AI in medicine |
| Computer Science, Stanford University | cs.stanford.edu | ontologies, knowledge representation, AI in medicine |
| Stanford Research Institute International | sri.com | applications of knowledge based systems |
| Computer Science, University of California at Los Angeles | cs.ucla.edu | uncertain reasoning |
| Information Sciences Institute, University of Southern California | isi.edu | knowledge representation, knowledge acquisition, AI in medicine |
| Microsoft research, Decision Theory and Adaptive Systems group | research.microsoft.com | uncertain reasoning, machine learning |
| Computer Science, University of Massachusetts | cs.umass.edu | empirical analysis of knowledge based systems |
| Computer Science, University of Rochester | cs.rochester.edu | agent systems, uncertain reasoning |
| Computer Science, University of Texas | cs.utexas.edu | knowledge representation, qualitative reasoning |

Table 2: Centres of excellence in the UK

| Place | URL | Themes |
|---|---|---|
| Department of Computing Science, Iniversity of Aberdeen | csd.abdn.ac.uk | knowledge acquisition, knowledge bases |
| Department of Computer Science, University of Wales at Aberystwyth | aber.ac.uk | qualitative reasoning, model based reasoning |
| Division of Informatics, University of Edinburgh | informatics.ed.ac.uk | knowledge representation, model based reasoning, agent systems |
| Department of Computing, Imperial College | doc.ic.ac.uk | knowledge representation, agent systems |
| Advanced Computation Research Laboratory, Imperial Cancer Research Fund | acl.icnet.uk | uncertain reasoning, agent systems |
| Knowledge Media Institute, The Open University | kmi.open.ac.uk | knowledge management, uncertain reasoning, web-based expert systems |
| Department of Computer Science, University of Manchester | cs.man.ac.uk | machine learning, case based reasoning |
| Department of Electronics and Computer Science, University of Southampton | ecs.soton.ac.uk | knowledge acquisition, agent systems |
| Department of Computer Science, University of York | cs.york.ac.uk | machine learning, agent systems |

Table 3: Other international centres of excellence

| Place | URL | Themes |
|---|---|---|
| Austrian Research Institute for Artificial Intelligence, Vienna | ai.univie.ac.at | knowledge representation |
| German Research Center for Artificial Intelligence, Saarbrucken | dfki.uni-sb.de | agent systems, communication |
| Division of Mathematics and Computer Science, Free University of Amsterdam | cs.vu.nl | knowledge representation, design lifecycles |
| Artificial Intelligence Research Institute, Barcelona | iiia.csic.es | agent systems, uncertain reasoning |
| Mizoguchi Lab, Department of Knowledge Systems, Osaka University | ei.sanken.osaka-u.ac.jp | knowledge representation, model based reasoning |

# 9 CONCLUSIONS

In general, safety has not been high on the list of priorities for expert system developers. Although there are some notable exceptions, primarily in the medical domain, most design in this area focuses on building systems rather than assessing or managing their safety. Yet, as we saw in Section 3, it is easy to find applications of expert systems in a variety of safety related application domains. Although currently safety related expert system design is a niche occupation it is also true that the use of expert systems (or similar technologies not traded under that lable) is increasing in general and there is no reason to expect that niches in safety related areas could not similarly expand.

In Section 4 we described the sorts of safety gains which might be obtained in a number of application areas where (as we established in Section 3) expert systems are already applied. These gains appear significant and would be difficult to achieve in these areas without the forms of technology which expert systems provide. Nevertheless, there remain difficulties in achieving safety for these systems and in demonstrating that it has been achieved. Section 5 raises potential hazards and Section 6 demonstrates problems in arguing a safety case for various kinds of expert systems. A key issue appears to be the way in which lifecycle and architecture fit together, since (as we saw in Section 2) the understanding of architectures for problem solving is well developed in the expert systems community, yet understanding of how these architectures and their associated lifecycles relate to safety analysis and management has been little explored. There are some hopeful signs of improvement, through initiatives such as SAFE-KBS and tools such as PROforma) but overall the outlook remains cloudy. Current trends suggest that this issue will become more important (Section 7) because there is a drive for more autonomous expert system related applications combining numerous different styles of design using both software and hardware components. If this happens then it will become more difficult to argue that expert systems are a special case, deserving either (on one extreme) to avoid disciplines of safety analysis and management or (on the other extreme) to be excluded from applications where safety is a concern.