Legal Aspects of Knowledge-Based Technology

Roger Clarke, *Department of Commerce, Australian National University, and Visiting Professor, Institut für Wirtschaftsinformatik, Universität Bern*

Abstract: A definition of knowledge-based technology (KBT) is provided which is more operational than conventional definitions of the term 'expert systems'. Ownership rights in products developed using KBT are considered and difficulties discussed. Legal liabilities which may arise from such products are considered and issues identified. It is concluded that commercial exploitation of KBT may be hindered by these legal difficulties. Some policy implications are identified.

Introduction

Expert systems have been an active area of research for three decades. In recent years claims have been made, not just by marketing interests but also by well-respected researchers, that they are graduating into a commercially exploitable technology (e.g. Michaelsen and Michie, 1983; Johnson, 1984; Blanning, 1985; Buchanan, 1986; Connell, 1987; Quinlan, 1987). Many large corporations and government agencies have been reported in the trade press as having embarked upon pilot projects, and some claim to have already implemented worthwhile applications.

Given that practical application is being made of a new and potentially very powerful technology, it is important that consideration be given to the legal framework within which future disputes regarding expert systems products and services will be resolved. There has been little discussion in the literature to date (but see Nycum et al., 1985; Zeide and Liebowitz, 1987).

This brief survey paper commences with a presentation of the nature of the technology, then discusses the legal frameworks of ownership rights, and of liabilities. Laws differ between countries, and the comments in this paper are phrased in a fairly abstract manner in an attempt to be applicable to at least UK, US and Australian jurisdictions.

Knowledge-based technology

The term 'expert system' is an unfortunate one, because of the wide range of interpretations to which it is subjected. In particular, the word 'expert' may refer to the source of the knowledge captured into the software, or the nature or standard of performance expected of it. The word 'system' may be understood very broadly, or may refer to a specific piece of software as in the term 'accounting system'.

For the purposes of this assessment, it is useful to use the more general term 'knowledge-based technology' (KBT). By this is meant *the application of a set of analytical and programming techniques and tools.*

Because the field is still so new, the set of techniques and tools is explained somewhat differently by leading texts (e.g. Barr and Feigenbaum, 1981, 1982; Hayes-Roth et al., 1983; Harmon and King, 1985). They may be classified broadly into three groups:

- *knowledge acquisition*, including rule induction and other machine-learning models;
- *knowledge representation*, including various models of semantic networks such as order-attribute-value triplets and frames, together with production rules, inheritance, plausible reasoning, and logic programming;
- *inference procedures*, including data-driven forward chaining and goal-directed backward chaining, depth-first and breadth-first search strategies, and non-monotonic reasoning.

Conventional software development technology deals with procedures or algorithms, which access precisely structured data, whereas KBT places the emphasis on 'knowledge'. Few authorities make clear what they mean by this term. In practice, the dominant manner in which it is used is as that which can be expressed in the form of antecedent-consequent-rules, i.e. IF-THEN-ELSE constructs. At least at the operational level, knowledge is used in contemporary KBT to mean sets of rules and heuristics pertaining to some problem domain, expressed as IF-THEN-ELSE constructs. Rules

are 'deep knowledge' based on causal models, whereas heuristics are 'surface knowledge' based on correlation alone. The important discontinuity, compared to conventional software development technology, is that the problem-solver no longer needs to think down at the level of the procedures and data which underlie the knowledge, and can therefore cope with more difficult problem domains.

The definition of knowledge used by KBT is regarded as dangerously narrow by some commentators, particularly from outside the information technology disciplines but also, to some extent, from within them (e.g. Dreyfus and Dreyfus, 1986a, 1986b; Roszak, 1986; Winograd and Flores, 1986).

Figure 1 provides a model of the development and use of KBT as it is currently practised. Users consult a 'knowledge base', by providing information about some event or situation within the problem domain. The software draws inferences, by applying the rules stored in the knowledge base to the case-specific data. A result is provided to the user in the form of a diagnosis, prognosis, recommendation, decision etc., depending on the nature of the application. In addition, an explanation may be provided showing the argument whereby the software reached its conclusion.

In order to establish the knowledge base on which the user consultation depends, knowledge is captured from one or more people, called 'domain specialists' (or 'experts'). This generally requires an intermediary analyst programmer referred to as a 'knowledge engineer', who expresses the knowledge using some appropriate language and supporting software.

The schema also incorporates three emergent areas of KBT:

- knowledge acquisition undertaken automatically, by analysing a set of historic cases, either to assist the knowledge engineer or (so far, less credibly) to create the knowledge base directly;
- general purpose knowledge bases (such as an encyclopaedia expressed in appropriate form) which may be used as a base upon which the domain-specific knowledge-base may be built;
- inherent learning ability, such that the results of new cases are used by the software to modify the existing knowledge base.

The term 'adviser' applications of KBT was coined some years ago to refer to software expressly designed to support human decision-makers rather than replace them. Adviser applications of KBT demand significant additional investment in user interface and explanation mechanisms. To distinguish software which actually makes decisions (e.g. in intelligent building environment and process control systems) this paper uses the term 'genuinely expert' applications of KBT. Of course, a poorly-designed adviser, or one used by a person inadequately trained or insufficiently sceptical of the tool, may adopt the status of a *de facto* genuinely expert application.

On the basis of these brief definitions and model, ownership rights relating to applications of knowledge-based technology are discussed.

Legal rights

Rights relating to software generally

Ownership rights in software are established through intellectual property law, the conventional term used to refer to a set of related areas of substantive law (Tapper, 1983; Ricketson). As far as computer software is concerned, the most important of these is copyright (Niblett, 1980; Lechter; Graham, 1984; Millard, 1985). The term also includes patents, (registered) designs, (registered) trade marks, trade secrets and *sui generis* (specific-purpose) approaches such as chip protection legislation. In different circumstances, each of these heads of intellectual property law has significance for software.

Remarkably, the applicability of copyright law to software has been uncertain until the 1980s. In the United States, it was established in a 1982 case that the 1980 amendment to the Copyright Act had successfully established that computer programs are copyrightable, and in a 1983 case that this is so irrespective of the form of the program (in particular source code, object code or ROM).

In the relevant Australian case, the courts decided that software was *not* copyrightable (Clarke, 1988). An amendment to the Copyright Act was rushed through Parliament in mid-1984, but has not yet been tested before the courts. In the United Kingdom, although legal commentators generally considered that software was copyrightable, the Copyright Act was amended in 1985 to make the coverage explicit. Neither the old nor the new provisions have been tested before the courts.

There are lingering doubts concerning works which originate in machine-readable form, since these may not be visible or otherwise humanly perceivable, as copyright law generally requires. Problems could arise in relation to design graphics, computer art, music and databases which come into being with the aid of computers. The same applies to text created using wordprocessing software and text-editors. In addition to poems, novels and business reports, software usually originates in this way and any such problem would therefore affect property rights in software.

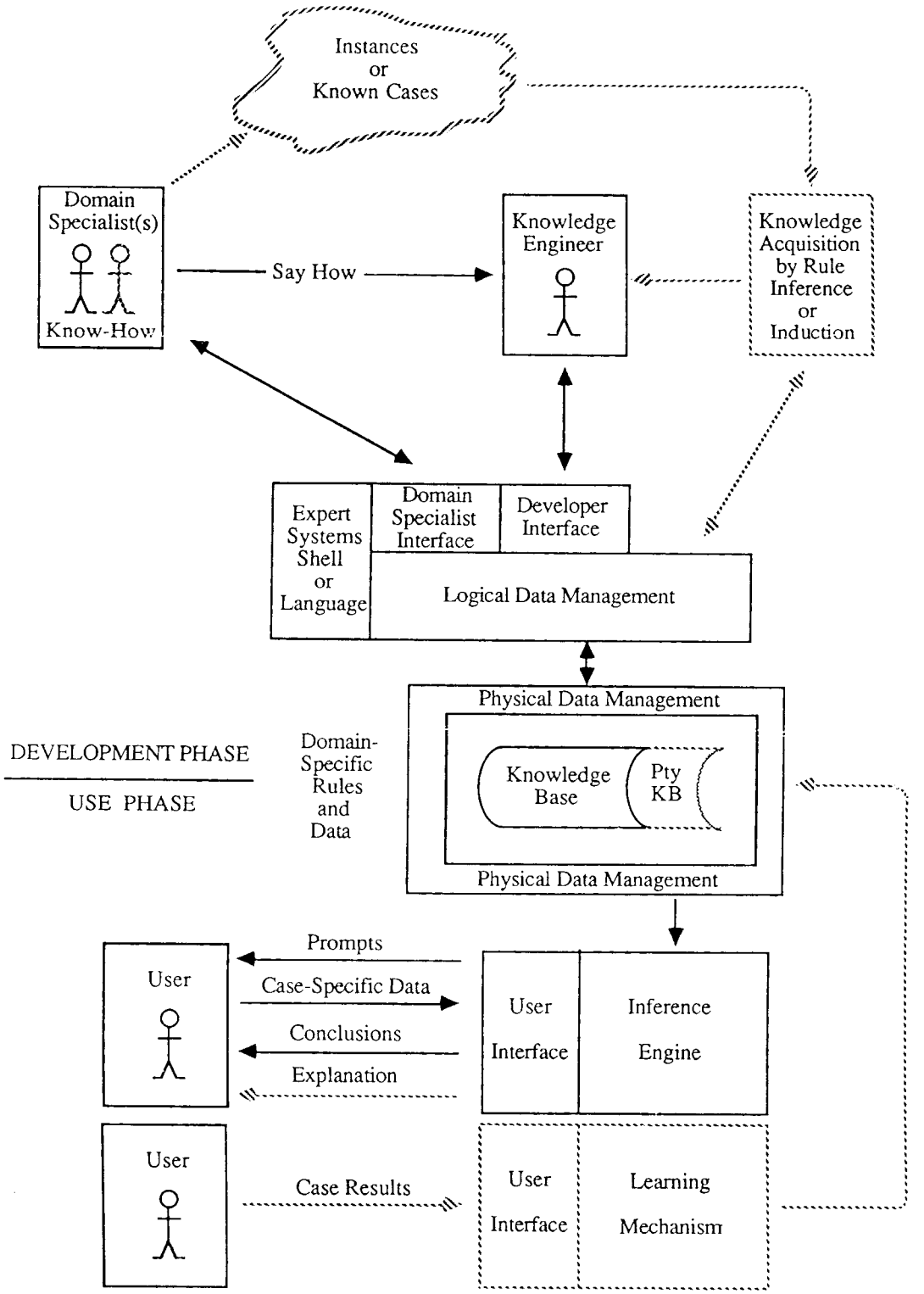

Figure 1. Basic schema for knowledge-based applications

Issues arising in relation to KBT

Where KBT is implemented using conventional languages and data management software, it probably enjoys the same level of protection as does any other software. However, KBT's aim is to break out of certain strictures placed upon the analyst/developer by conventional development infrastructures. A knowledge base is not 'software' in the conventional sense, but rather a set of interrelated rules, expressed in a manner that enables processing by a particular kind of run-time interpreter called an 'inference engine'. It is not clear that courts will construe knowledge-bases to be software for the purposes of copyright. They might treat them as some other type of work to which copyright applies, or they might find them to be uncopyrightable.

There are further aspects to the question of ownership rights in software. KBT pilot projects have been undertaken in a number of organizations to capture the expertise of a person who is about to be lost to them, typically due to retirement. Until now, these projects have been perceived very positively by management, and have involved the services of a highly-regarded knowledge engineer. The behavioural phenomenon popularly called the 'fishbowl effect' has ensured the goodwill of the domain specialist. Consider the same project in the absence of the aura of importance and novelty. Where the domain specialist declines the invitation to express his know-how in understandable terms, by what means can an employer seek to enforce his will and his moral (and presumed legal) right? Laws relating to trade secrets, confidence and Official Secrets may preclude the person from divulging the knowledge to another. Employment law might preclude the person from using the knowledge to the advantage of himself or a future employer. It may provide an enforceable right to the employer regarding the divulging to him of specific information or data (e.g. what was said in a telephone conversation with a client). However, it appears very unlikely that the law provides employers with anything approaching an intellectual property right in the knowledge of their employees.

Where an application or program generator is used to generate software, who owns the result – the person on whose behalf the parameters were supplied, or the person who owns the program-generator? This 'meta-authorship' problem (Hofstadter, 1979) arises in automated knowledge acquisition, where 'surface' (correlation-based) rules are generated from cases, and entered into a knowledge base. Are these rules owned by the organization for whom the induction process was performed, the owner of the copyright in the cases (if any exists), or the owner of the induction software? The problem becomes even more difficult to resolve in the case of machine learning, since the knowledge base could be substantially changed by the software self-modifying a single rule.

Legal liabilities

Liabilities arising in relation to software generally

Legal responsibilities in relation to computer software generally can arise in a variety of ways (Nycum, 1979; Gemignani, 1981; Rumbelow, 1981; Tapper, 1983, pp.75-95 and 212-215; Parry, 1983; Scott, 1984; Smedinghoff, 1986; Edwards and Savage, 1986; Brown, 1986).

The most common source of liability is where a *breach of a contractual warranty* results in harm to a party to a contract (e.g. Cheshire and Fifoot, 1981; Guest, 1984). *Express warranties* are part of the contract, and may arise from descriptions of software, expressions of fact or promises important to the transaction, or software demonstrations. In addition, an *implied warranty of 'merchantability'* exists if the provider of a product is generally in the business of providing such products. This requires that a product be fit for the ordinary purpose for which such a product is used. The *implied warranty of 'fitness for (specific) purpose'* imposes a higher standard, but only arises in certain circumstances. A huge body of case law embodies many exceptions and interpretations. In some jurisdictions there are additional layers of statute law, particularly in transactions involving consumers.

Liabilities can also arise in relation to computer software as a result of *fraudulent misrepresentation or deceit.* A representation can be a written or oral statement, can be implied by conduct (including a product demonstration), or might be implied by silence.

Every person has an obligation to take 'that degree of care, precaution and vigilance which the circumstances justly demand', and if he fails to do so, may be liable in *negligence* for personal injury and property damage. The standard of care inferred by courts depends on the status of the person in relation to the activity he is performing, with professionals expected to demonstrate a higher standard of care.

In the UK and Australia, and especially in the USA, forms of *no-fault liability* are tending to be imposed, both by statute (such as Sale of Goods and Trade Practices Acts) and by common law. Such 'product liability' (sometimes called 'strict liability') applies to goods but generally not to services, and to sales but generally not to licences. Hence it may apply to 'turnkey systems', and perhaps

Quality of use
- decision when and when not to use any particular KBT application;
- need for user dialect and domain understanding consistent with those of the domain specialist and knowledge engineer;
- non-visibility of defects;
- blind administration of machine-dictated decisions seriously exacerbated;
- method of ensuring detection of the need for mainteance;
- difficulty of maintenance.

Quality of product
- lack of an adequate scientific basis;
- over-simplification of inherently difficult and complex matters;
- non-transmissibility of techniques
- the rigours of financial administration;
- experimental, prototyping approach;
- independence from corporate database and transaction processing.

Moral and legal responsibility
- high level of abstraction of software development activity, hence absence of explicit definitions of solutions or even problems;
- increased ambiguity of responsibility;
- absence of responsibility for learned, autonomous behaviour.

Table 1. Liability issues arising in relation to KBT

to some kinds of system software sold with a machine. Otherwise it has very limited applicability to software, and probably none at all to services such as contract programming and consulting. However, that position could change very rapidly.

In addition, a range of *other torts* may give rise to liabilities in some circumstances. An example would be defamation arising from a false statement (e.g. concerning a person's credit record or creditworthiness) issued 'automatically' by a computer (i.e. under program control, without human intervention).

Several of the areas of law discussed above apply only to the extent that software is deemed by the law to be a product. Generally, custom-built software and software created by customizing packages are unlikely to be treated as products. However, standard application packages, systems software (especially if it is delivered with a computer), and application software delivered as part of a 'turnkey contract' are more likely to be deemed to be, or to be part of, a product. There are few jurisdictions where such matters have been clearly established.

A problem shared with other complex technologies is that information technology products involve the co-operation of several suppliers, and hence the assignment of liability among suppliers can be difficult.

Although the laws relating to responsibility for the actions of human agents are reasonably well understood, those relating to the delegation of responsibility to machines are less clear. In general, the use of a computer cannot shield a person from his obligation to exercise due care (Nimmer and Krauthaus, 1986). However, some circumstances may arise in which the courts may find it unreasonable to sheet legal responsibility home to the user organization.

Issues arising in relation to KBT

Many of the issues discussed in this section are of some concern in the case of adviser applications, and of serious concern in the case of 'genuinely expert', decision-making applications. Table 1 summarizes the issues.

A number of elements combine to cause concern about the *quality of use* of KBT applications. Users must decide when and when not to use any particular KBT application, and they therefore need an appreciation of its scope and the boundaries of applicability. Then, to use it effectively, they need a dialect and domain understanding consistent with those of the domain specialist and knowledge engineer. This requires significant investment in user education and training and the user interface. Adviser applications require even more investment in the user interface, and in explanation mechanisms.

Because defects in KBT applications are difficult to detect, suspect applications may continue to be used. Their rationale is even more mysterious to the end user than that of conventional software, and hence the existing incidence of blind administration of machine-dictated decisions could be seriously exacerbated.

Maintenance of KBT applications represents a further area of concern. The need for modifications to cope with changing external

circumstances is more difficult to detect than with conventional software. There are also greater difficulties in performing maintenance, because the programmer cannot foresee the impact that a new or amended rule will have on the whole.

Organizations which use KBT-based applications may lay themselves open to actions in negligence and other torts such as defamation, unless they take appropriate, additional precautions regarding education and training, choice as to when to use the software, knowledge-base maintenance, and review of automated and semi-automated decisions.

A variety of factors result in concerns about *quality of product*. Some critics within the computer science community claim that KBT lacks an adequate scientific basis (e.g. Hofstadter, 1985). This applies particularly to non-deterministic forms of KBT, an area commonly referred to as plausible reasoning (e.g. O'Neill, 1987). The pseudo-mathematics of Mycin's 'certainty factors' are a case in point.

Because of this inadequate basis, there is a real danger of over-simplification of inherently difficult and complex matters. Available models or formalisms may be imposed on problem domains, because of their convenience to the knowledge engineer rather than their suitability for representation of the particular type of knowledge. Related risks are an excessive reliance on correlation-based heuristics rather than causal model-based rules (surface rather than deep knowledge), and the cramming of 'grey' areas of knowledge into deterministic models. The danger exists of losing the ambiguity and tolerance needed by human systems to cope with unforeseen and extenuating circumstances, ambiguity of legal and moral prescriptions, and (increasingly rapid) changes in social standards.

The popularization of any technology involves its simplification, the training of practitioners with experience in earlier technologies but without an appropriate educational base, and application in many different real-word contexts. Techniques used and understood by pioneer knowledge engineers might not be transmissible to more ordinary mortals.

It is normal to adopt an experimental prototyping approach to KBT applications. Clearly, quality assurance is more difficult in such circumstances than with more planned and disciplined software engineering methods. As KBT is commercialized, the ethos of the pioneers will have to be moderated, and methods adapted to provide greater confidence in the outcome. However, another effect of the routinization of KBT will be that fringe modules such as audit trails, user interfaces and explanation subsystems (despite their importance noted in the previous section) are likely to be among the first casualties of budget pruning.

A further area of concern is that KBT-based applications have generally been conceived as independent software, often running on equipment with limited connectability with the organization's major equipment. For KBT applications to reflect up-to-date 'knowledge', they must be integrated with corporate database and transaction processing systems.

As a result of product deficiencies, user organizations may incur additional liabilities in negligence, or in other torts such as defamation. In addition, the supplier (or any and all members of a chain of supplier organizations) may be subject to additional contingent liabilities – in some cases to user organizations, in others directly to an affected third party. These may readily arise under contract law, fraudulent misrepresentation and negligence. If the no-fault liability area of law were to develop quickly, it would be very easy for KBT applications, by their very nature, to be interpreted by the courts as being in a 'defective' or inherently dangerous state when they left the original supplier.

Moral and legal responsibility in relation to KBT is also a matter of concern. The generations of conventional software development technology have successively decreased the extent to which the human analyst/developer has needed to understand and define the problem and its solution. The third generation of procedural languages enabled programmers to express a problem solution in a convenient form. The current fourth generation of program and application generators enables the human to delegate the solution to the computer, and focus on the problem definition.

KBT models not the problem, but the problem domain. It relieves the software developer of the responsibility for formulating explicit definitions of the problems which are to be solved. If they exist at all, the problem definitions are neither within the program, nor the knowledge base, nor the case-specific data, nor the inference engine. Each problem definition is implicit within a particular process or case, which is an ephemeral combination of elements of all of them.

KBT brings software development to the point where the analyst/developer no longer needs to have a clear understanding of the problems the product is to solve. The moral and professional responsibility of developers is therefore diminished. More remotely, should users become heavily reliant on KBT-based software, professional standards among domain specialists could decline – for example among provincial and country medical specialists.

At the same time, there is an increased ambiguity of legal responsibility. The user can invoke the 'piano-player's defence', i.e. 'It was my job to apply the tool, not to understand it.' However, the developer can conceivably adopt the same stance since his role was merely to capture the know-how of another person or people into machine-processable form. Meanwhile the domain specialist(s) can avoid responsibility because the form in which the knowledge was expressed only vaguely resembled their knowledge, and they could not be expected to understand and audit the particular formalism used by the knowledge engineer.

Automated knowledge acquisition based on correlation rather than causality further risks diminution of responsibility. And assigning legal responsibility for learned, autonomous behaviour will be a serious challenge to the law.

It seems likely that harm arising from KBT applications may not be effectively justiciable. In some cases this will be because no person can be held legally responsible for mitigating the harm done; in others because determination of responsibility is dependent upon an unbearably lengthy (i.e. five- to ten-year), expensive, energy-sapping and legal-technical test case.

Implications

Because the law is always uncertain, and ever (slowly) changing, the pioneers of any new technology face some degree of risk that the law will not recognize their presumed protections, or will retrospectively impose unexpected responsibilities, and hence liabilities. Some of the risks faced by early adopters of KBT are that knowledge bases may not be copyrightable; that knowledge bases could be deemed to be products, and hence implied warranties and product liability applied to them; that the courts could interpret professional standards to have developed sufficiently quickly that a high standard of due care applies to software products and services; and that a slow and expensive appeals process will have to be pursued in order to establish the law.

Management will be well advised to recognize the importance of deciding whether an application is an adviser, or a 'genuinely expert' application of KBT. Adviser systems will require significantly greater investment in user interface and explanation subsystems.

There are also considerable implications for governments. If the benefits of KBT are to be realized, at the same time as ensuring public protection, key uncertainties in the law should be removed. Copyright in knowledge bases should be clarified. The legal definition of 'product' should be amended to explicitly include software and knowledge bases, or explicitly exclude them, or include them in explicitly defined circumstances (e.g. when sold with hardware). If restrictions on the sharing of court costs were eased, the law could be clarified without a first transgressor having to carry all the arrows in his back. In court cases in which difficult technological matters are a central issue, the courts could be encouraged, or empowered, to adopt more effective techniques. The present approach using expert witnesses supplied by litigant and respondent could be replaced or supplemented by the judge having his own technical advisers, in some sense at least, on the bench.

Where it is appropriate to impose the risk of new products on their purveyors, self-regulation and risk-sharing should be encouraged. These arrangements might take the form of mutual insurance funds, or compulsory third party insurance as in workers' compensation and motor vehicle accident insurance.

There are enormous gaps and uncertainties in public liability law, in relation to both negligence and accident. Computer software is less potentially devastating than nuclear energy and the chemical, petrochemical and biochemical industries. If it fulfils its promise, through such developments as knowledge-based technology, that may not always be the case.

References

Barr, A. and Feigenbaum, E.A. (eds.) (1981, 1982) *The Handbook of Artificial Intelligence* Vols 1 and 2. Kaufman.

Blanning, R.W. (1985) Expert systems for management: research and applications. *Journal of Information Science*, 9, 153-62.

Brown, C.M. (1986) Liability for the supply of defective software. *Computer Law and Practice*, 3, 1, 2-12.

Buchanan, B.G. (1986) Expert systems: working systems and the research literature. *Expert Systems*, 3, 1, 32-51, January.

Cheshire, G.C. and Fifoot, C.H.S. (1981) *The Law of Contract.*

Clarke, R. (1988) Judicial understanding of information technology: the case of the wombat ROMs. *Computer Journal*, 31, 1, 25-33, February.

Connell, N.A.D. (1987) Expert systems in accountancy: a review of some recent applications. *Accounting and Business Review*, 77, 67, Summer.

Dreyfus, H.L. and Dreyfus, S.E. (1986a) *Mind Over Machine.* Blackwell.

Dreyfus, H.L. and Dreyfus, S.E. (1986b) Why expert systems do not exhibit expertise. *IEEE Expert*, Summer.

Edwards, C. and Savage, N. (eds.) (1986) *Information Technology and the Law*. Macmillan. (See especially the articles by Chalton (pp.5-24) and Kaye (pp.25-51).)

Gemignani, M.C. (1981) Product liability and software. *Rutgers Journal of Computers, Technology and Law*, 8, 173-204.

Graham, R.L. (1984) The legal protection of computer software. *Comm ACM*, 27, 5, 422-6.

Guest, A.G. (ed.) (1984) *Anson's Law of Contract* 26th edition.

Harmon, P. and King, D. (1985) *Expert Systems*. Wiley.

Hayes-Roth, F. et al. (1983) *Building Expert Systems*. Addison-Wesley.

Hofstadter, D.R. (1979) *Godel, Escher, Bach: an Eternal Golden Braid*. Penguin, 1980, p.607. (Originally published by Basic Books, 1979.)

Hofstadter, D.R. (1985) *Metamagical Themas*. Penguin, 1986. (Originally published by Basic Books, 1985.)

Johnson, T. (1984) *The Commercial Application of Expert System Technology*. Ovum, London.

Lechter, M.A. (1981) Protecting software and firmware developments. *IEEE Computer, pp. 73-82*.

Michaelsen, R. and Michie, D. (1983) Expert systems in business. *Datamation*, November.

Millard, C.J. (1985) *Legal Protection of Computer Programs and Data*. Sweet and Maxwell.

Niblett, B. (1980) *Legal Protection of Computer Programs*. Oyez.

Nimmer, R. and Krauthaus, P.A. (1986) Computer error and user liability risk. *Defense Law Journal*, 35, 4, 579-99.

Nycum, S.H. (1979) Liability for malfunction of a computer program. *Rutgers Journal of Computers, Technology and Law*, 7, 1-22.

Nycum, S.H. et al. (1985) Artificial intelligence and certain resulting legal issues. *The Computer Lawyer*, pp.1-10, May.

O'Neill, J. (1987) Plausible reasoning. *Australian Computer Journal*, 19, 1, February.

Parry, A.E. (1983) Data processing risks – seven examples of exposure. *Computer and Society*, 13, 3, Summer.

Quinlan, J.F. (ed.) (1987) *Applications of Expert Systems*. Addison-Wesley.

Ricketson, S. (1984) *The Law of Intellectual Property*. Butterworths.

Roszak, T. (1986) *The Cult of Information*. Pantheon, USA.

Rumbelow, C. (1981) Liability for programming errors. *The International Business Lawyer*, 9, 303.

Scott, M.D. (1984) *Computer Law*. Wiley.

Smedinghoff, T.J. (1986) *The Legal Guide to Developing, Protecting and Marketing Software*. Wiley.

Tapper, C. (1983) *Computer Law* 3rd edition. Longman.

Winograd, T. and Flores, F. (1986) *Understanding Computers and Cognition*. Ablex.

Zeide, J.S. and Liebowitz, J. (1987) Using expert systems: the legal perspective. *IEEE Expert*, pp.19-22, Spring.

Biographical notes

Roger Clarke joined the Department of Commerce at the Australian National University in 1984 as Reader in Information Systems. This followed 17 years in professional, managerial and consulting work in the private sector in Sydney, London and Zürich. His areas of interest are application software technology and its management, and economic, legal and social aspects of information technology. This paper was completed during a Visiting Professorship at the Institut für Wirtschaftsinformatik at the University of Bern, Switzerland.

Address for correspondence: Department of Commerce, Australian National University, Canberra ACT 2611, Australia.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.