# Introduction

This text follows on from a number of insightful works in the Cambridge Studies in Intellectual Property Rights series, and I have set my goals to achieve as useful a work as have my predecessors – only the reader will know how close to that goal I have managed to get. However, there is a significant difference between this work on intellectual property and the others in the series, in that those others were much more firmly located in substantive and comparative law issues. This text differs somewhat and the difference can be found in the dual nature of my academic career – first as a computer scientist and then as an academic lawyer, but always with a highly sociological bent – and is directed at trying to match the new and radical technology of computing with the patent system, rather than provide an overview of substantive and comparative law: there are a number of other texts which already provide a legal overview very effectively.[1] If there is novelty in this project, it is in the attempt to move the frame of reference so that the patenting of software 'as such' is seen as a respectable and valid goal, but is also seen as a project in which there is much to do.

My aim has also been to encourage computer scientists to engage with patent law, rather than – as many I suspect would prefer – to have lawyers just leave them alone. This latter option is no longer available: software is being protected, but will not be protected in the most appropriate manner unless there is involvement from the field of computer science.

I began work on this project under an ESRC research grant[2] with quite a neutral perspective on the value of software patents, indeed perhaps – as with many academics with an interest in IP matters – slightly sceptical of their utility. The project as originally conceived was, basically, to browse

[1] For example, K. Beresford, *Patenting Software Under the European Patent Convention* (London: Sweet & Maxwell, 2000) provides a patent attorney perspective, and I. Lloyd, *Information Technology Law*, 4th edn (Oxford: Oxford University Press, 2004) provides a technically literate approach to the legislation and case law.
[2] RES-000–22–0158.

through the published patent documentation and try to determine the kinds of tactics being employed to gain protection and whether there were methods of more-properly defining what should be allowable than recourse to the concept of 'technical contribution'. As I browsed and my recall of computing and its loci of interest grew, I found more and more that I could not really see in the documentation the inventiveness which I knew could be found in computing – there was plenty of software invention but always hidden away in a manner which undermined that inventiveness rather than affirming it. My thinking about software patents changed and the conclusions I drew became more positive, though with some reservations.

This text is directed, therefore, towards the conceptual space between computing and law, rather than being a substantive or comparative law analysis. Few lawyers really understand the technology of software, and many technologists simply see lawyers as the problem rather than the solution. This has led to, on the one hand, software creation being viewed by lawyers as 'mere data processing' and on the other hand to patent law being viewed as highly undesirable by programmers. This interdisciplinary area is difficult, requiring handling of both computing and legal concepts and an understanding of their interaction. Patent law in the other fields – engineering, electronics and chemistry, for example – has been well developed and in fact the patent system grew at the same time as these fields themselves grew. But with the new technology of software we have a radical technology being fitted into a well-established body of law, with perhaps only limited success. This has meant that, rather than this being limpid prose, there may be a more dense nature to the writing than I would have liked. The reader has my sympathy, but it does seem essential that if lawyers are to consider this new technology they have to see it as the developers themselves see it. There is recent evidence of just how much effect the legal world can have on technology and why a proper understanding is necessary: large sums of money and time were spent on 'solving' the Year 2000 problem after lawyers began to raise the possible issues of negligence. Some will suggest that it was only through lawyers highlighting the issue that there were no actual problems at the stroke of the year 2000, but the view from many companies was different, and many senior IT managers were sacked for having wasted company resources on a hype promoted by lawyers and their ilk. The debate – now mostly forgotten – in the academic legal literature about privacy and Caller ID appears to me to be another such aspect of the confusion found when conjoining lawyers and technology. Hopefully, this text will be useful to those interested in technology and law, rather than only to the patent lawyer.

Not only is the approach in this text focused more towards software technology than pure legal analysis, it is also influenced by a procedural approach to law: that is, that often procedure and practice are more important elements in understanding law than the bare rules. This approach was one which I developed during research into the barrister's profession with a colleague and published as *The Barrister's World and the Nature of Law*.[3] In that research we were struck by just how important it was to the lawyer to know procedural mechanisms and that they were more often prouder of their procedural knowledge than their substantive law knowledge – one interviewee suggesting that if you cited more than a couple of authorities to a county court judge he would 'just switch off'. The move from researching the law profession to patent law was something of an accident – a research interview with an IP barrister about developing the barrister research towards looking more at the specialist bar led me to suggest that the European Patent Office might be a good alternative location for a research project. This led to that interviewee suggesting that I shouldn't bother with the EPO since it wasn't very interesting. I temporarily gave up plans for further study of the UK IP profession[4] and took up the challenge, reported in the text *Harmonisation of Intellectual Property in Europe: A Case Study in Patent Procedure*,[5] which was essentially a socio-legal study of European Patent Office procedure and practice. The barrister who suggested that I ignore the EPO was Robin Jacob QC, and the reader of the *Harmonisation* text may agree that his advice should have been taken: but I would hope that it might aid understanding of where the potential examination problems with software may lie.

The EPO research was funded by the Research Fund of the European Patent Organisation and further funding was made available to study the Boards of Appeal. This appeared as 'Judicial and Administrative Roles: The Patent Appellate System in a European Context'.[6] Once again, my

[3] P. Leith and J. Morison, *The Barrister's World and the Nature of Law* (Milton Keynes and Philadelphia: Sweet & Maxwell, 1992).
[4] This was undertaken but the results have not yet been published.
[5] *Vol. 3, Perspectives on Intellectual Property* (London: Sweet & Maxwell, 1998. Note that some changes have occurred since this text was published. For example, DG1 (search) and DG2 (examination) have combined into DG1, and DG2 now deals with operations. There is also a level of unhappiness amongst EPO staff which was not found when I carried out this research – pressures to increase workload apparently causing a strike: 'If we have to produce more, then patent quality will go down', [Wolfgang Manntz, the chair of the Berlin branch of the EPO staff union] said. 'We are already at the limits of working productivity.' Reported at http://news.zdnet.co.uk, 10 May 2006.
[6] *Intellectual Property Quarterly* 1 (2001), 50–99.

research was influenced by Jacob J, who had, in *Lenzing*,[7] suggested that: '[t]he fact is that the members [of the Boards of Appeal] are independent in their judicial function and that independence is guaranteed by the EPC itself. They are judges in all but name – and it is rather a pity that they were not so-called by the Convention.' This statement coloured my research in a number of ways. My conclusions were slightly different, suggesting that there was an 'engineering' mentality in the boards and sometimes a conflict between legally qualified members and technical members. This is an interpretation which has influenced my argument here, particularly in Chapter 1. It would be untrue to suggest that my findings were accepted by the boards themselves – quite the contrary.[8]

This asynchronous relationship – as computing might describe it – with Jacob LJ (as he is now) continued during the writing of this text as I awaited the decisions in *Aerotel* and *Macrossan*. These were handed down with sufficient time to incorporate them into the text and Chapters 5 and 7 make recourse to them.

What is the argument I am putting forward? In essential terms:

- It quickly became obvious to the boards of appeal (that is, to Board 3.5.1) that the software exclusion under the EPC was not practical. It was not practical because software was becoming a major part of all areas of technology.
- Having what might be called an 'engineering approach', they felt that there was a technical framework which would bypass the Art. 52 exclusion. This was that programs which were part of/related to physical devices were not software 'as such'. This is the solution of a practical community rather than a legal concept which explains the difficulty courts have had with it.

These points are dealt with in Chapter 1. Chapter 2 moves towards a more software-inclined perspective and argues:

- that the creativity and inventiveness of the programmer is being substantially undervalued by the patent system;
- that there is a more appropriate way to understand invention in software than that of the machine-oriented approach; but
- that the malleability and the descriptive techniques used in software mean that description (and thus patentability) can be difficult.

[7] *Lenzing AG's European Patent (UK)* [1997] RPC 245.

[8] I presume they were unhappy with, for example, the reporting of their comments, such as one chairman who suggested: 'Most lawyers I work with wouldn't know the top end of a machine from the other end. How can they assess technical aspects?' and from a lawyer: 'Some [technical] colleagues are a bit afraid of the more abstract/intellectual approach, but that's a minority.' The research highlighted a tension between legal and technical approaches. This is important in helping to understand the nature of 'technical effect'.

Chapter 3 outlines in brief the policy context of software patentability, arguing that it is difficult to prove the need or not for protection for software. Chapter 4 uses a number of patent examples which are accessible to lawyers to highlight that simply removing the software exclusion would not be sufficient to remove the tensions in the system: that software is such a different technology that it requires a better/different examination than is currently being given to it by patent offices.

In Chapter 5 the problem of trying to draw a line in the sand is raised, and I suggest that such lines are difficult to make. Indeed, it seems to me that the technical contribution approach is more amenable to allowing protection of business methods than pure software. In Chapter 6 the possibility of alternative forms of protection are discussed, with the conclusion that these are essentially 'utility model' approaches and best seen as a complement rather than an alternative to patent protection. Chapter 7 draws together these arguments and looks towards possible mechanisms for change. I argue – surely to the objection of software patent opponents – that protection 'as such' will arrive and that, for computer science as a discipline, it may be advantageous.

The approach as a whole is thus pro-software patent, but aware of the problems which the system has and will continue to have in handling such a radical technology. It is also one which suggests that it is better to deal with these problems in the open than to pretend that they do not exist.

In terms of terminology, I use 'programmer'. This can be considered out of date and many in commercial computing now prefer terms such as 'software engineer' but, as I have argued elsewhere (in *Formalism in AI and Computer Science*[9]), there are problems with using an 'engineering' descriptor. 'Programmer' has become almost a term of abuse (alongside 'data processing') and I think this would have dismayed the early innovators in the field who were forever battling against the perceived importance of hardware and insignificance of software. Even though the balance has changed and software design has demonstrated its inherent difficulty, those involved in software still appear to have too humble a view of their role, and using the term 'engineer' – in my view – only highlights this lack of confidence.

My thanks to George Woods for helpful comments on Chapter 4.

Finally, where appropriate I use the term 'he' for both 'he' and 'she'.[10]

[9] London and New York: Ellis Horwood/Simon and Schuster, 1990.

[10] Since completion of the manuscript, T0154/04 from B.A 3.5.1 has become available to me. It responded to the Aerotel/Macrossan judgment discussed in Chapter 7. As predicted in that chapter, the Board of Appeal does not view its decisions as requiring a referral to the Enlarged Board.

# 1 Software as machine

*We sense that we know 'technology' when we see it. And no doubt that is correct, most of the time. But it is not correct all of the time. Therein lies the delusion. You can prove that for yourself by trying to find a definition of 'technology' that everybody can agree on. The more you try, the more you will discover what a horribly imprecise concept it is.*[1]

## The problem: invention and the definition of technology

A prediction: within the next decade or so it will be possible to gain patent protection for software in the widest sense across all of Europe. Another prediction: algorithms which are not tied to any specific computer implementation will be *openly* protectable, as will business methods.

A prediction that software will be protectable is hardly adventurous, since, as Beresford[2] has argued, such patents have been granted by the European Patent Office for some years now. Beresford's thesis is that it was only a general misconception which led to a belief that 'computer-implemented inventions'[3] were not protectable: he pointed out that a reading of the EPO's annual report from 1994 noted that 11,000 such patents had been granted and only 100 refused. Now, a large number of patents – which are best described as 'software patents' – are entering the national phases of European EPO member states[4] in a variety of technical fields.[5]

[1] Peter Prescott QC sitting as Deputy Judge in *Patent Applications by CFPH LLC* [2005] EWHC 1589 (Pat).
[2] K. Beresford, *Patenting Software Under the European Patent Convention* (London: Sweet and Maxwell, 2000).
[3] This is the preferred term of the various patent offices, etc. It will be used interchangeably with 'software patent' and 'software-related invention'. Part of the author's argument will be that it can also frequently be used interchangeably with 'business method patent'.
[4] See a review of current developments in *Emerging Technologies: the EPO approach: EPO Seminar on Search and Documentation Working Methods*, EPO briefs – collected works series (The Hague: EPO, 2005), available online at http://www.european-patent-office.org/dg1/searchseminar/2005/_pdf/sfa_2005_103_giannotti.pdf.
[5] There is no single 'software' classification at the EPO and neither did the EPO have a classification similar to the US 'business method' Class 705 (Data Processing: Financial, Business Practice, Management, or Cost/Price Determination) until the 2006 revision of the International Patent Classification.

The prediction that algorithms and business methods, too, will be openly protectable – by which I mean that patent offices will no longer suggest that such protection is not available in Europe – is perhaps a more debatable point: yet, here too, we see indications that protection is being given for 'inventions' which differ substantially from the traditional form. For example, the inventive element in claim 1 of Menashe's 'Interactive, computerised gaming system with remote terminals' (EP625760) is clearly a 'gaming system' rather than any novel application of the underlying technology: the basic idea in the patent is a centralised host computer with individuals interacting with this system via home computers, where 'aspects of the invention concern auditing and security to ensure fairness for players and prevent players defeating the outcome of a game'. The application date for this patent was 1994, a date at which a considerable amount of research work on networking had been carried out, including auditing and security issues and it is clear from the specification that the novelty in the invention is located in computerised gaming: the prior art cited in the application by the inventor was primarily gaming systems.[6] The patent specification includes two diagrams, one being a representation of a home computer and the other outlining the hardware elements of the invention. The simplicity of the technical framework in this patent can be seen from these two diagrams. In the diagram (Fig. 1.1) reprinted here (Fig. 1 in the patent), 11 refers to the system being part of a wider network of computers, 12 is the central host, 14 refers to nodes 'constructed in a known manner to monitor the flow of data' (the networking hardware is modem based). 18 is the player's terminal, 22 refers to links to local terminals (presumably for local players) and elements 26 refer to 'data bases of player, game and accounting information as well as programs for the host and for downloading to the terminals when required'. The novelty of the system principally appears to be that the home computer runs one part of the gaming system via a program and the remaining part is run on the central server.

The system is an example of a 'computer-implemented invention' – without computerisation we could imagine that a board game might be constructed with some elements being carried out by a 'banker' and some by the remaining group of players. However, with the addition of program control there develops a different situation: the speed of processing, the locational divergence of players and the ability for users to play different games simultaneously all lead to an artefact which radically differs from

[6] For example, EP0542664 'Electronic system for the controlled play of bingo and machines usable with the system', which was granted, but after opposition was revoked through lack of inventive step.

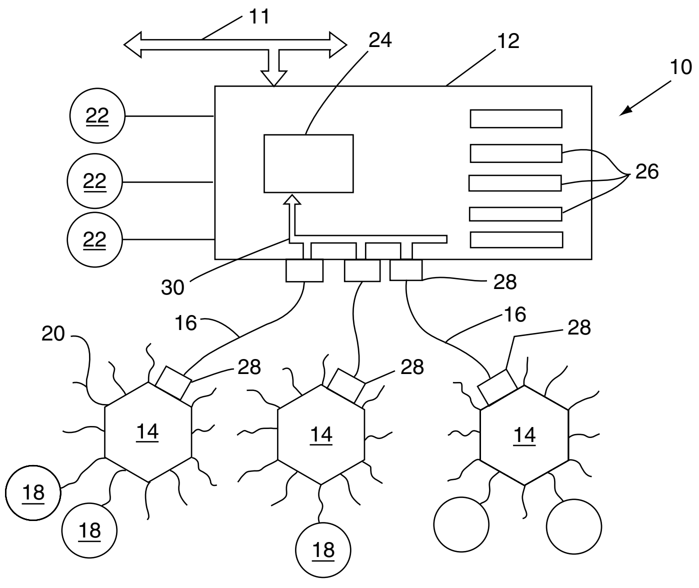

Fig. 1.1. The Menashe hardware

that based upon a non-computer-based system such as the board game, Monopoly®. Without being drawn into argument over whether the Menashe patent should have been granted at all – e.g. was it 'inventive enough' – we can see that it can be read to show one of the current problems at the heart of the European patenting system and the heated debate over software, algorithm and business method patents: new technology allows applications to be developed which do not substantially differ from the non-computer-based underlying implementations of the idea except insofar as they would not be useful if they were not computerised. We have to decide whether they should be protected because they are advances in technology (gaming technology, perhaps) or should they be denied because their use of existing hardware and software technology is simply routine.

The argument I wish to present in this chapter is that the underlying problem at the heart of European software patenting has been the debate over the meaning of 'machine' rather than any specific legal definition of 'technical effect' or suchlike – that is, how the notion of this technology has been constructed by courts and patent offices from its early years of development. I will suggest that older models have been used and little attention has been paid to the internal perspectives of the software community. Since the legal community has spent some 20 years attempting to pin down the meaning of 'technical effect' in a legalistic definition and has – many would argue, to date – been unsuccessful,[7] there may some utility in standing back and using a different theoretical approach.

Also part of my argument is that the important decisions were made in the 1980s. Board of Appeal 3.5.1 of the European Patent Office – which was given the workload for the relevant classifications – clearly took the view that the role of a patent office was to give protection to technological developments and that since such developments were happening in software control of hardware, they should be protected. Once that step was taken – that applications should no longer be denied simply because they involved the use of a program – the path was laid out: any attempt to hold a line becomes untenable because the definition of protectable technology changes under the continual assault of perceptive patent attorneys who locate logical contradiction and push the examiners towards removing that logical weakness. That early position, where all applications for software-related inventions were denied, for example, could not be held because any good patent attorney with relevant understanding could transmute a software invention into a hardware one, thus undermining the line which the examiners were attempting to hold. Protection may have been less with that form of claim, but it may have provided sufficient defence to be worthwhile.

This argument is hardly surprising or particularly novel to those with an interest in technology and related sciences: we have seen similar processes take place in arguments over definitions of mathematical objects.[8] Such processes are a widespread part of the human condition[9]

[7] Bertil Hjelm, Principal Director DG2, suggested (in 2001) that the lack of a definition of 'technical' may actually be a benefit since allowing the Boards of Appeal to 'adapt it to unanticipated future technologies is a major strength of the EPC. The price paid must be a lack of certainty in borderline cases as to what will and will not be accepted.' WIPO/ECTK/SOF/01/2.4. Of course, neither of the two concepts used by the Boards of Appeal – 'technical effect' and 'technical contribution' – is derived from the EPC at all, so can hardly be one of its major strengths. Not only do the patent professionals have difficulties with these concepts, but so do system users. See the report on the 'UK Patent Office Technical Contribution Workshops', *CIPA Journal* (April 2005), 251–3, where the observer noted: 'it was difficult to have people with little experience of claim and legal construction apply the definitions and really consider the actual meaning of, or even notice, many of the limitations in the definitions of technical contribution.'

[8] I. Lakatos, *Proofs and Refutations* (Cambridge: Cambridge University Press, 1976) shows this with respect to polyhedron through a conversational technique. Mathematical histories also show these developments with, for example, the debates over whether a set could include other sets of infinite size.

[9] Anthropologists have been concerned about the nature of 'boundary' for many years. Even the discipline of anthropology has caught itself up in boundary problems – see G. W. Stocking, 'Delimiting anthropology: historical reflections on the boundaries of a boundless discipline', *Social Research* (Winter 1995), available online at http://www.encyclopedia.com/doc/1G1-18229219.html.

and indeed appear in the arriving at the meaning of legal concepts, too. In the patent field we see that the 'manner of new manufacture' requirement in UK law was a derivation from the earlier technological notion underpinning the Statute of Monopolies,[10] which was used to encourage new manufacturing techniques in the seventeenth century and was pushed somewhat to cover the new technologies of, for example, 'vendible products'.[11]

Of wider interest is the legal discussion of the nature of software itself: someone with a background in computer science would be struck by the often simplistic approach which lawyers take to software.[12] It is similar to that which those in computing take to law: they view it as a relatively coherent and stable system of knowledge which evidences a system of clear and lucid rules.

Despite the fact that this is not entirely a text on the policy decision of whether software should be protected or not, given that there has been such a heated debate over the Directive for Computer-implemented Inventions,[13] one is usually expected to take a position on whether software patents should be allowed or denied. The position outlined here is that their introduction was inevitable, but that from a perspective of general industrial policy it is difficult to see a rationale which allows inflatable kayaks to be protected but a major element of industrial production not to be so protected. Such developments as business method protections will certainly cause substantial problems for some in the software and other related digital industries – which are outlined in the following chapters – but there may also be some positive benefits arising from the pressures which the new patenting world will bring: a raised level of inventive step; a better and cheaper system of patent litigation than we have been used to in the UK; the development of a more 'scientific' approach in computer

[10] See S. Thorley, R. Miller, G. Burkill and C. Birss, *Terrell on the Law of Patents* (London: Sweet & Maxwell, 2000), §§1.11, 2.01–2.06.

[11] *Re GEC's Application* (1942) 60 RPC 1.

[12] Neitzke, when writing in the 1980s about the lack of clarity in US judgments, suggested: 'But nothing in the decisions suggest that the [Supreme] Court has any real appreciation of how a program is created' or that the Supreme Court judges 'would recognize a source program on sight': F. W. Neitzke, *A Software Law Primer* (New York: Van Nostrand Reinhold, 1984). In attempting to mould their client's interests to the legal framework, the attorneys may not have helped them with an accurate picture of programming or software. It is not just judges who seem to fail to get what programming is about: many commentators also fail – see, e.g., G. Ghidini and E. Arezzo, 'Patent and Copyright Paradigms vis-à-vis Derivative Innovation' *IIC* 36 (2005), 159, where they get part of the idea, but do not get the whole idea: '. . . software programs remain codes not readable by human beings.'

[13] Directive on the Patentability of Computer-implemented Inventions, COM(2002) 92 Final.