Cambridge Intellectual Property Rights and Information Law

Software and Patents in Europe

Philip Leith

CAMBRIDGE

CAMBRIDGE www.cambridge.org/9780521868396

This page intentionally left blank

Software and Patents in Europe

The computer program exclusion from Article 52 of the European Patent Convention (EPC) proved impossible to uphold as industry moved over to digital technology, and the Boards of Appeal of the European Patent Organisation (EPO) felt emboldened to circumvent the EPC in *Vicom* by creating the legal fiction of 'technical effect'. This 'engineer's solution' emphasised that protection should be available for a device, a situation which has led to software and business methods being protected throughout Europe when the form of application, rather than the substance, is acceptable.

Since the Article 52 exclusion has effectively vanished, it is timely to reconsider what makes examination of software invention difficult and what leads to such energetic opposition to protecting inventive activity in the software field. Leith advocates a more programming-centric approach, which recognises that software examination requires different strategies from that of other technical fields.

PHILIP LEITH is Professor of Law at The Queen's University of Belfast.

Cambridge Intellectual Property and Information Law

As its economic potential has rapidly expanded, intellectual property has become a subject of front-rank legal importance. *Cambridge Intellectual Property and Information Law* is a series of monograph studies of major current issues in intellectual property. Each volume contains a mix of international, European, comparative and national law, making this a highly significant series for practitioners, judges and academic researchers in many countries.

A list of books in the series can be found at the end of this volume.

Software and Patents in Europe

Philip Leith

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK
Published in the United States of America by Cambridge University Press, New York
www.cambridge.org
Information on this title: www.cambridge.org/9780521868396

First published in print format 2007

ISBN-13 978-0-511-36636-9 eBook (EBL)
ISBN-10 0-511-36636-1 eBook (EBL)

ISBN-13 978-0-521-86839-6 hardback
ISBN-10 0-521-86839-4 hardback

For Christine and Annie

Contents

List of figures

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.

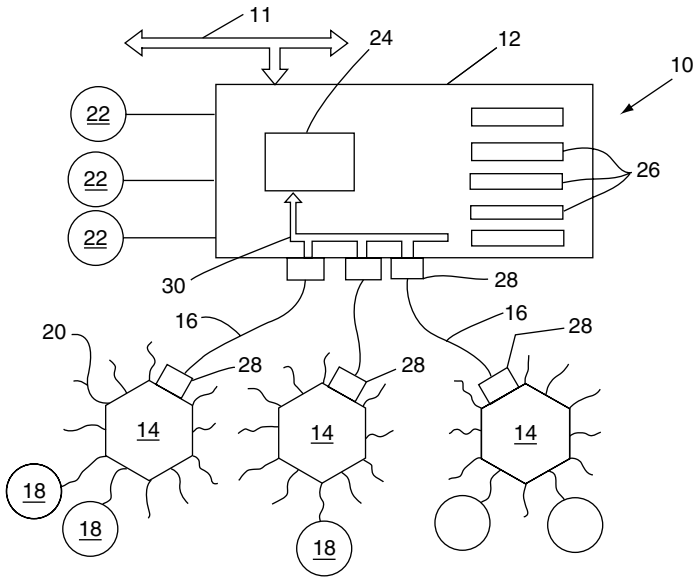

Fig. 1.1. The Menashe hardware

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.

'science'; the integration of monopoly-based approaches more directly into patent law; a divorcing of patent appeal from patent granting.

The critics of software patentability who argue that software is 'new', and requires special treatment, tend to forget that many of the technical problems which are of interest now have already been met in some form or other over the past 50 years of development. It would be surprising if this was not the case – computing has been a significant economic enterprise since the 1960s and many of the major technical solutions upon which computing is now based had begun to be developed by or during that period. For example, Unix was developed in the 1960s and by the early 1970s was being used (on a PDP machine with *roff* software) to write patent applications by Bell Labs.[14] Machines were slow, storage was expensive and input/output was hardly up to our current standards, but the basic processing operations, such as multi-tasking (where more than one task is carried out at any one time), allocation of resources by round robin (giving each user a fraction of time before moving onto the next user) and other processes were well understood. There had been more than sufficient significant technical advance to seek patent protection for these ideas but many of the earliest patent applications were for hardware and much litigation therefore related to hardware.[15]

How did we get here?

Prior to the inception of the European Patent Convention, the patenting of software and also what are now called business method patents were valid in the UK. Patent law prior to the EPC made no specific reference to software and thus applications were dealt with as general technology and had to meet the requirements of the 1949 Patent Act as to patentability – that is, that the invention related to a 'manner of new manufacture'.[16] This broad-based definition could clearly include software within its remit and, indeed, we find that in the 1970s there were attempts to move from the solely hardware-oriented patent application to that of application-oriented patent, where the novelty lay in the software being run on non-novel hardware.

[14] D. M. Ritchie, 'The Evolution of the Unix Time-sharing System', in *AT&T Bell Laboratories Technical Journal* 63(6) (October 1984), 1577–93.

[15] See, e.g., the argument over validity of the ENIAC patent discussed in A. R. Burks and A. W. Burks, *The First Electronic Computer: the Atanasoff Story* (Ann Arbor, MI: University of Michigan Press, 1989).

[16] Section 101(1), Patents Act 1949: ' "invention" means any manner of new manufacture the subject of letters patent and grant of privilege within section six of the Statute of Monopolies and any new method or process of testing applicable to the improvement or control of manufacture, and includes an alleged invention.'

As a useful example of this move we can examine the Nymeyer patent (GB1352742), which was filed in 1971 from an earlier US application,[17] entitled 'Improvements Relating to Data Processing', which in many ways – particularly in that it was an application to protect a business method – was a precursor to the Signature patent[18] litigated in *State Street*,[19] a decision which has revolutionised the patent system in the US. The specification outlines that the invention related to the use of a computer to operate a market:

> The present invention relates to data processing, and has particular application to 'fungible goods', hereby defined for the purposes of the present specification as anything which is available in quantity and of which one unit is precisely equivalent to another unit so that a buyer will not care which particular unit he obtains, for example commodities, stocks and shares, in an auction market, hereby defined for the purposes of the present specification as a market in which fungible goods are bought and sold, for example a commodity exchange or stock exchange. For the purposes of the present specification 'at market' orders are hereby defined as orders to buy or sell at the prevailing market price which is unspecified in the order.

Claim 1 detailed a 'data handling system suitable for establishing prices for a given kind of fungible goods'. The general similarity to the Signature patent can be seen from the Signature description:

> A data processing system is provided for monitoring and recording the information flow and data, and making all calculations, necessary for maintaining a partnership portfolio and partner fund (Hub and Spoke) financial services configuration. In particular, the data processing system makes a daily allocation of assets of two or more funds (Spokes) that are invested in a portfolio (Hub). The data processing system determines the percentage share (allocation ratio) that each fund has in the portfolio, while taking into consideration daily changes both in the value of the portfolio's investment securities and in the amount of each fund's assets.

Both of these patents are essentially implementing business ideas via a computer system – Nymeyer was setting up an auction system to allow individuals to buy and sell shares, while Signature is similarly

[17] See the US equivalent, US 3,581,072 'Auction Market Computation System' and note the change in title for the UK office. Filed 1968.

[18] See US 5,193,056 'Data processing system for hub and spoke financial services configuration'. A PCT application was filed to include Europe (WO9215953) but the application was later deemed withdrawn without examination. The radical step of *State Street* was effectively to make the only requirement for patentability to be that the application covered something 'useful'. For a more detailed coverage of the current US situation, see G. Stobbs, *Business Method Patents* (New York: Aspen, 2002).

[19] *State Street Bank & Trust Company* v. *Signature Financial Group, Inc.*, 149 F 3d 1368 (Fed. Cir. 1998).

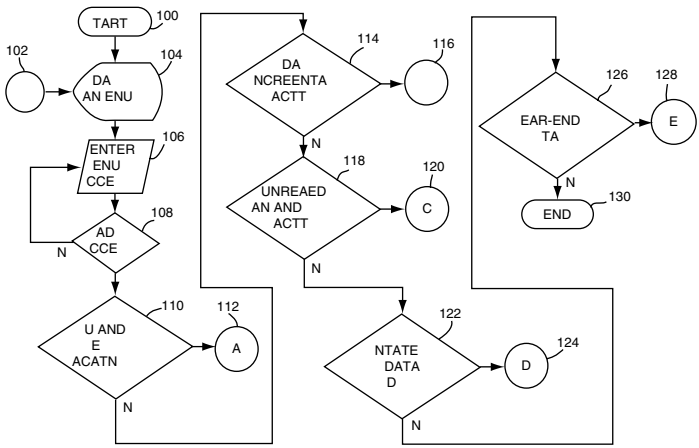

Fig. 1.2. Signature's hub and spoke flowchart

purchasing and selling shares but doing so by using a networked system to combine the purchasing power of a partnership. Neither of these patents utilise 'novel' computing techniques yet they both demonstrate different approaches to describing the invention and to both can be applied the conclusion of Whitford J: 'Such an operation could in theory be done without the need for any automatic aids but in practice needs to be automatically computed.'[20] Signature certainly uses a flowchart methodology to represent this (Fig. 1.2) but it is a flowchart which shows a business process rather than a detailed program implementation.

Nymeyer lays out the business method diagrammatically (Fig. 1.3) as a mixture of process, software and hardware – as, indeed, 'a machine'.

Images are important in the description of software – as we see below in discussion of *ideograms*[21] – since they determine how an artefact which is mutable is presented. Here, Nymeyer's patent attorney was clearly melding together various integers to produce what he hoped would be viewed as a single inventive entity. For example, the 'Price Determine Gate' is described in circuit logic terms:

[20] *Application by IBM for the Revocation of Letters Patent No. 1,352,742 In the Name of Frederick Nymeyer*, Patents Appeal Tribunal, 16 October 1978. The text of this judgment was added as an appendix to a brief *amicus curiae* for Chevron Research Company in *Sidney A. Diamond* v. *Diehr* 450 US 175 (1981) and is available online.

[21] L. Fleck, *Genesis and Development of a Scientific Fact* (Chicago, IL: Chicago University Press, 1979), p. 137 defines these as: '... graphic representations of certain ideas and certain meanings. It involves a kind of comprehending where the meaning is represented as a property of the object illustrated.'

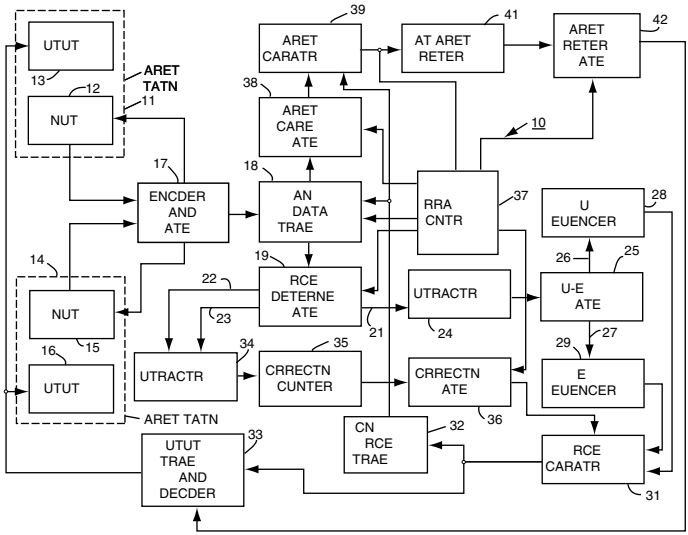

Fig. 1.3. Nymeyer's combination of hardware, software and market terminology

The main data storage unit 18 is provided with an output circuit that is connected to a price determining gate circuit 19 having three outputs 21, 22, and 23. The output circuit of 21 of gate 19 is connected to a subtractor circuit 24.

A description using such terms would not have been usual amongst either hardware designers or programmers of the 1970s: it was a descriptive picture drawn up by the patent attorney to emphasise the machine-like qualities of the invention. In effect, it was a composite structural image which – when actually programmed – would have been entirely different in both software and hardware form, being run on a general-purpose computer and programmed in Fortran, Cobol or something similar.

The application was successful and the UK patent was granted in 1974. IBM objected to the grant and requested revocation, which was heard by Graham J and Whitford J in the Patents Appeal Tribunal. The only point argued between the parties was whether the invention was an invention under the 1949 Act[22] – that is, whether it was a 'manner of new manufacture'. The court thus looked only at claim 1, since if that failed the patent as a whole failed. IBM's primary objection was that the invention covered a standard computer and therefore could not be directed to a manner of new manufacture. Given that the specification outlined such

[22] The new 1977 Act was not relevant.

elements as a comparator, if the claim had limited itself to a computer with such special hardware, IBM's objection would not be valid but – in IBM's view – since the claim did not limit itself to such a special purpose computer and one of their standard computers would (if programmed to do so) be capable of storing the relevant data, making the necessary comparison and selecting the appropriate price, this broader claim was unpatentable. In the view of IBM, even if the idea is new and not obvious, 'it does not become a manner of manufacture if you do no more than call in the aid of an ordinary computer to do the work for you'.

The court, however, took a different perspective – one which has much more in common with recent European Patent Office positions: it looked at the invention as a whole, and sought similarities with the traditional 'machine'. This is not to say that they were duped by the nature of the invention – they clearly saw that the inventive element was a business scheme[23] – but they were prepared to use the machine analogy to understand the invention; that, when programmed, the general purpose computer becomes a special purpose machine (an 'apparatus'), even if no special hardware is evident in the final system:

> As matters stand, however, until Mr Nymeyer came along there was no reason to suppose that anyone would have thought of writing the appropriate programme and building it into a computer or otherwise putting it into a physical form suitable for use with a standard computer. A computer programmed to carry out Mr Nymeyer's system must we think be considered as being an apparatus having novel characteristics.

The UK patent office and courts were generally in this period agreeable to the notion that a new machine came into being when a standard computer was programmed. There was little concern with the idea of the program itself being protectable rather than the system as a whole: none of the decisions raised this as a possible area of concern. In large part this was simply due to the technology at the time: machines were expensive, software was usually packaged with the hardware, and the marketplace for software was radically different from that of today.

What are we to make of Nymeyer's patent? Clearly, it was a business scheme of the order which has now become controversial within Europe. Yet the court did not view this as problematic and had little problem in accepting that this was a 'manner of new manufacture' and integrated it under the rubric of 'machine', following the manner of seeing that if cam control of lathes is protectable (a well-developed technology of the 1970s

[23] 'We agree that it is plain that this patent derives from an idea Mr Nymeyer conceived as to the best way of determining a selling price for, for example, a stock or share . . .'

involving physical control by wheels with cut-outs on their circumference prior to CNC manufacture) then programs were simply the computer equivalent. They did distinguish that this patent was not a mathematical algorithm, which – if they had found to be the case – may have significantly altered their outlook.

As with all judgments, there is very little contextual information available on why the patent came about or what has happened since. What did Nymeyer intend for the patent? It was not worked commercially since electronic trading in stocks did not begin until many years after the patent had been issued. We tend to assume that all patents are applied for in the spirit of seeking financial reward (the philosophical justification for the system) and it may be that in many cases this is so. However, software can differ from this quite substantially, since it has a 'theoretic' component which can underpin philosophical or ideological positions. Although I have been unable to find anything written by Nymeyer on his patent application (despite the fact that he was quite a prolific writer) it can be seen that there was more than a financial agenda behind his application. Nymeyer was a successful businessman and an active supporter of the 'Austrian school' of economics which was based upon the methods of Ludwig von Mises – a classical liberal in economic thought. Nymeyer was a founder of the Libertarian Press, an economics publisher which continues to publish classical libertarian texts and was also involved in religious organisations promoting Dutch Calvinism. The aim of the patent seemingly was to push the idea of citizen involvement in share dealing via technology and undermine the professionalised control of the marketplace in shares. Market prices, he felt, should be freed from the siphoning off of profits, and prices should be free to fluctuate wildly without the artificial constraints added by specialists.[24] The US patent specification indeed references an extract from a reprint of an economics work which was published by his Libertarian Press.[25] Though considered a machine by a court, it was certainly a politically-inspired machine – pushing Nymeyer's view that: 'It is that financial calculation in terms of gold which controls the thinking of all well-informed people.'

[24] This information was provided by J. Sprowl, who recalled discussions some years before with the author of Nymeyer's patent application (Walther E. Wyss).

[25] E. Von Boehm-Bawerk 'Value and Price', pp. 215–35 (an extract from the three-volume work *Capital and Interest* (South Holland, IL: Libertarian Press, 1960). Nymeyer's views have been reported as: '[h]is main contention was that our race resolutions abridged his Christian freedom to discriminate. He claimed the divine right to disapprove of bow legs, crooked teeth, and black skin, and did not wish to have his liberty judged by another man's conscience.' (See H. Stob at http://www.stobfamily.com.)

The physicality of the invention – that is, its representation as machine rather than program – is one which was common in the early UK applications. Beresford, in his *Patenting Software*, has discussed this process as the making of 'something tangible' when he discusses the construction of the claims in *Badger Co. Inc's Application*.[26] The attorney had first claimed: 'The method of mechanically designing and forming a visible drawing illustrating a piping system . . .', which was rejected by the patent office, since this was essentially a process of designing a plan, which clearly it is, and that the –

> mechanism required, i.e. computer and data converter or plotter, are known instruments requiring no further description or illustration, so that the only potential novelty which the specification discloses resides in the manner in which the operation of these machines is conducted . . .

Lloyd-Jacob J noted the parallel for designing and forming to be found in a chemical process: 'the forming being regarded as the selection and preparation of the raw material for the process and the second as the provision of reaction conditions necessary to secure the required conversion.' However, the analogy broke down because there were no physical elements being combined as in a chemical process, since the 'raw material is conceptual in character, lacking the concrete actuality which differentiates substance from notion, which actuality is not necessarily conferred by writing, printing or otherwise representing the intellectual information upon a sheet, card, tape or other carrier'. The problem in *Badger* was that the components as set out in the application did 'not contribute to the fashioning of some product such as the definition of invention has hitherto been thought to require'. Thus there was an invention, it was patentable, but it was not being correctly claimed in the application, and this correct format was that of a machine and a process for 'conditioning' that machine. The application was remitted back to the patent office with the clear direction of the judge that, if claim 1 was reworded towards a machine format, it would be appropriate. The amended claim thus became: 'A process for conditioning the operation of a computer and associated plotter . . .', with no amendment required to the steps in the method which had been laid out in the original application (i.e. the core of the invention).

We can see that the machine analogy was central to the early success of applications in the UK: in *Slee and Harris's Applications*,[27] too, the same

[26] [1970] RPC 36; [1969] FSR 474. The original US Patent was titled 'Automated Designing' (US 3,636,328), filed 1968. A continuation was granted as US 3,867,616.
[27] [1966] RPC 194.

approach was taken – that a claim to a method of operating a computer was not allowable, but a claim to a computer when modified to operate according to the method and a claim to a means of controlling the computer were allowed. This did not mean that a new machine *per se* was required, simply that if the invention was abstract it was unpatentable, but if linked to the control of hardware then it was patentable. The only requirement from the patent attorney wishing for successful prosecution of the application was to ensure that a physical machine was somewhere at the heart of the claimed invention. The distinction seems hardly more than semantic but it was more than that – it was attempting to fit the new technology of software into an already-existing mental conception and the best-fitting conception was that of machine.

In the US similar developments in software technology were happening: the US Patent and Trademark Office was easing its opposition towards the protection of what was intrinsically a software invention, though the early applications always covered themselves with the cloak of device. Pal Asija claims, in *How to Protect Computer Programs*, to have prosecuted the 'first pure software patent'[28]. This related to 'Swift Answer' a 'system of full text, free-form, narrative, information input, storage and retrieval', which allowed the user to ask free-form questions of the system in query language. The application was filed in 1974, at a time when there was much interest in information systems, such as legal databases where developers had discovered that, to be successful, they were required to hold the entire text of a judgment rather than just a short abstract. Given the difficulty of using these systems to full effect, improvements in querying and recall would have been substantial advances, with the patent's aim being: 'to facilitate retrieval of narrative textual information by making the system very forgiving of user's mistakes with respect to rules of programming, punctuation, syntax, grammar and spelling, etc.'[29] Still, claim 1 of the patent, rather than being directed purely towards software, utilised the machine framework: 'A process for information input using electromechanical devices, storage, search and output comprising the steps of . . .' Thus, though this was a process claim and hence the basis of the argument that this was a 'pure software patent' – rather than a product or machine claim, for example – we see that the process in claim 1 entails some hardware components, perhaps because patent prosecution without

[28] US 4,270,182. Of course, as with anything technical, there are disputes about invention. Stobbs early suggestion included a 1972 patent, US 3,633,176, or even a 1951 patent, US 2,552,629, if not Samuel Morse's 1840 patents on signal processing ('dots and spaces'). G. A. Stobbs, 'Information wants to be free, but the packaging is going to cost you!', *Mich. Telecomm. Tech. L. Rev.* 75 (1996).

[29] P. 152.

locating the hardware at all would have been impossible to get past the examiner. It would take the *State Street* decision to move discussion away from the machine entirely to the notion of 'usefulness'. But at the time the patent at the heart of the *State Street* case was being examined,[30] US applications were still making reliance upon hardware dressing: for example, the patent was called 'A data processing system ...' and claim 1 referred to 'A data processing system for managing a financial services configuration ...' Prior to *State Street*, the change in US approach had meant that a software invention could be protected when it was claimed in the form of a process as well as – like the Badger 'Automatic Designing' patent – a 'system', so long as that system was machine-like.

The significance of *State Street* for the diversion of paths between the US and Europe on patentability cannot be underestimated – since 'usefulness' is now the primary requirement for patentability in the US, an 'anything goes' attitude appears to have developed and many observers have their favourite examples of patents which demonstrate this. One is Olson's 'Method of Swinging on a Swing':[31] 'A method of swing [sic] on a swing is disclosed, in which a user positioned on a standard swing suspended by two chains from a substantially horizontal tree branch induces side to side motion by pulling alternately on one chain and then the other',[32] an application which could – one might hope – hardly succeed in Europe.

Setting current US developments aside, however, the major distinction between the US approach and the UK one prior to the implementation of the EPC was that US patent law allowed only four categories of invention – process; machine; article of manufacture; and composition of matter and software – many fitted easily into the process and machine classifications. In the UK, the need to locate it legally led to a best fit under the concept of 'manner of new manufacture', with the bonding of novel software and well-known hardware as an inventive 'machine'.

It is interesting to note that both the Nymeyer patent and the Badger patent related to products which were later to become central to their two fields. For example, a world of share dealing without electronic trading would be unimaginable – with the technology first being applied within the stock-broking profession itself and now open to home investors. Nymeyer's invention preceded the major impact of producing a global

[30] US 5,193,056, 'Data Processing System for Hub and Spoke Financial Services Configuration', filed 1991. The Euro PCT application (WO9215953) based on this US filing was deemed by the EPO to have been withdrawn in 1998.

[31] US 6,368,227.

[32] The advance in usefulness claimed is that the older, traditional forms of swinging back and forward 'can lose their appeal with age and experience. A new method of swinging on a swing would therefore represent an advance of great significant and value'.

share-dealing network and also the more minor: the introduction of e-commerce techniques leading to a new profession – that of day trader – who utilises the speed of computerised share dealing to buy and sell on the same day, hopefully with large profits, just as he wished. Even less imaginable would be a world of engineering without computer-aided design. During the early period of commercial development, the CAD market grew from under $25 million in 1970 to just under $1 billion by 1979[33] and is now well over $100 billion in value. CAD software has developed further, with tools being designed to integrate with the designed software so that computer-aided manufacture (CAM) is possible, and the entire engineering chain has been reworked to encompass supply chain integration based upon these CAD/CAM techniques, allowing globalisation of industrial production with design occurring in one or more locations and supply, manufacture and delivery being arranged from any number of other locations.

Neither of these developments required software implementation as an essential: it would be perfectly possible to produce special purpose machinery utilising logic circuits which carried out exactly the same function as that produced by a standard computer under program control, but the costs involved in doing this would be considerably higher because software development – for all its many problems – is still a more efficient and effective way of producing these artefacts than through electronic circuitry which was and remains commonly open to patent protection.

By 1969, the UK patent office had arrived at a position which continued until the implementation of the 1977 Patents Act:

> Patents are not granted for computer programmes expressed as such. No objection is, however, raised in respect of inventions for novel methods of programming computers to operate in a specified way, or for computers so programmed, or for tape etc. having recorded on it a novel programme to control a computer in a stated way. Nor, in general, is objection taken to inventions involving new uses of computer in controlling manufacturing processes or to methods of testing, involving novel programmes, for computers under manufacture.[34]

It was a position based upon a legal fiction – that software and hardware became a new machine. Many commentators noted the semantic sleight of hand. In the Banks review, for example:

> The position on program patents seems to be that they are granted by the patent office when the claims are drafted so that they relate to a piece of machinery or apparatus e.g. a computer when programmed in a certain way; a tape or card with a certain configuration of holes; a computer-controlled process such as a power station

[33] http://www.cadazz.com. [34] UK Patent Office, February 1969.

or steel works provided that the invention claimed shows novelty. Whether or not there can be said to be a real distinction between a program invention claimed in this rather roundabout way and a claim to the program itself is at least open to doubt.[35]

Germany, too, had taken a relatively liberal approach (based upon the 'technical' model which later found its way into EPO thinking) with *BCD-Conversion*.[36] However, such a liberal view of patenting was not to be found in all of Europe.[37] France, for example, specifically excluded programs or sets of instructions by statute.[38] Why might there have been such a difference in approaches to the technology? One reason may well be the historical development of the technology. In the UK theoretical work on computability had been carried out by Alan Turing; Maurice Wilkes and colleagues produced the first stored program machine (EDSAC) at Cambridge; and what has been claimed to have been the first 'office computer' system was installed at Lyons Teashops in February 1954.[39] Programming, too, had been part of the technological development. EDSAC had, for example, developed a subroutine library of commonly used programming routines of which Dijkstra has suggested: 'We should recognise the closed subroutines *as one of the greatest software inventions*; it has survived three generations of computers and it will survive a few more, because it caters for the implementation of one of our basic patterns of abstraction.'[40] Wilkes has pointed out that one of the first programs to be run on EDSAC was a game program – a vertical line of dots on the screen represented a fence and a user could – by placing his hand in the light beam of the photo-electric paper tape reader – cause a hole to be placed in that fence in different locations. A horizontal line of dots would then periodically move slowly towards the right and, if the hole was in the correct location in the fence, the dots would pass through; otherwise not. Further, the program would try to detect a pattern in the user's placement of the hole in the fence and, if it did, the dots would always get through the hole.

[35] M. A. L. Banks (Chairman), *The British Patent System: Report of the Committee to Examine the Patent System and Patent Law* (London: Stationery Office, 1970), Cmnd. 4407, para. 474. (hereafter the Banks Committee Report on Reform of the Patent System, 1970).

[36] *BCD Conversion (In re Western Electric Co. Inc.)* IIC vol. 5, No. 2, 1974, pp. 211–16 (Federal Patent Court, 17th Senate, 1973).

[37] G. Kolle, 'The Patentability of Computer Software in Europe and under International Patent Treaties', in H. Brett and L. Parry (eds.), *The Legal Protection of Computer Software* Oxford: ESC Publishing, 1981) provides a round-up of the various attitudes taken by the member states of the EPC.

[38] Article 7, para. 2(3) of the Patent Act 1968.

[39] G. Ferry, *A Computer Called Leo* (London: Fourth Estate, 2003). Note that the Leo system was part hardware, part software and part business method re-engineering process.

[40] E. W. Dijkstra, 'ACM Turing Lecture: The Humble Programmer', *Commun. ACM* 15(10) (1972), 859–66 (emphasis added).

As Wilkes suggests: 'No one took this program very seriously, but it contained the germs of computer graphics, man-machine interaction, and artificial intelligence.'[41] He should also have noted that his was a precursor for what has become a major industry – computer gaming.[42] The computer games industry is worth, in the US alone, $7 billion.[43]

Intellectual property policy has always had a strong national element – anyone who attends meetings organised by the various industrial and commercial bodies will see that attendees always argue for a legal position (usually involving more protection) which might protect their own local industry. Jacob pointed to this in his criticism of the growth of industrial property.[44] Thus countries like, for example, France, with little indigenous software industry, would be expected to prefer software not to be protected since it would be of little benefit to their much smaller industry – a situation which in some senses continues: even in 2006, it was being reported that 'you could put the entire French software industry into Symantec, which alone generates about €3B in annual sales'.[45]

The EPC exemptions

After much negotiation among the potential member states and the failure of the Community Patent which was to have brought the European patenting system under the control of Europe and the Court of Justice, the 'temporary' European Patent Convention (EPC) was signed in 1973. The European Patent Office began to take applications in June 1978 and, though there were worries that it might not succeed, it became a highly successful operation, undermining the workload of the national offices of the member states.

The negotiation around the EPC had included discussion of what should constitute a patentable invention, the result being Art. 52 and its list of exemptions. Pila has provided[46] an analysis of the changing positions of the various countries as the draft EPC was firmed up, and there appears to be little in the way of logical development of the ideas: the liberal wing and the

[41] M. Wilkes, *Memoirs of a Computer Pioneer* (Cambridge, MA: MIT Press, 1985), p. 208.
[42] The first patent for a computer game was US 2,455,992, issued on 14 December 1948.
[43] 2006 Figure produced by Entertainment Software Association, at http://www.theesa.com.
[44] R. Jacob, 'Industrial Property – Industry's Enemy', *Intellectual Property Quarterly* 1 (1997), 3.
[45] Reported 27 April 2006, http://www.thealarmclock.com.
[46] See J. Pila, 'Dispute over the meaning of "Invention" in Art. 52(2) – The Patentability of Computer-Implemented Inventions in Europe', *IIC* 36 (2005), 173 and J. Pila, 'Article 52(2) of the Convention on the Grant of European Patents: What Did the Framers Intend? A Study of the Travaux Preparatoires', 36 *IIC* (2005), 755.

more conservative Europeans both believing that they had arrived at an EPC position which reflected their current national position. For example, the UK must clearly have seen the use of 'as such' as simply following their own current situation as laid out in the Patent Office note of 1969 (above). Kolle has suggested, too, that a significant element in deciding how to handle software was the Patent Co-operation Treaty (PCT). This was already in existence as an international searching mechanism and some of the offices which were already examining under PCT may thus have felt confident in following this approach. The finally agreed text was:

Patentable inventions

(1) European patents shall be granted for any inventions which are susceptible of industrial application, which are new and which involve an inventive step.
(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1:
  (a) discoveries, scientific theories and mathematical methods;
  (b) aesthetic creations;
  (c) schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;
  (d) presentations of information.
(3) The provisions of paragraph 2 shall exclude patentability of the subject-matter or activities referred to in that provision only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such.
(4) Methods for treatment of the human or animal body by surgery or therapy and diagnostic methods practised on the human or animal body shall not be regarded as inventions which are susceptible of industrial application within the meaning of paragraph 1. This provision shall not apply to products, in particular substances or compositions, for use in any of these methods.

Despite the success in achieving the negotiated Article, we might say that much of the history of the EPC has been the attempt to circumvent the exemptions set out in Art. 52. Such negotiations towards agreement and then later interpretative attempts to unravel the process of negotiation for an underlying set of principles are not unusual in law generally or patent law in particular: compare the insertion of agreed propositions pertaining to protection and promotion of the useful arts and sciences into the US Constitution[47] and the current debate over extent of

[47] See I. B. Cohen, *Science and the Founding Fathers: Science in the Political Thought of Thomas Jefferson, Benjamin Franklin, John Adams and James Madison* New York: Norton, 1995). Article 1: 'To promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries.'

protection arising from the constitution in terms of 'anything under the sun that is made by man'.[48]

This text is concerned with those exemptions in Art. 52(2) and (3) but it is useful to note that section (4) has also been of concern, with various Enlarged Board of Appeal decisions[49] laying out just what is exempted and what is not with regard to therapy. The exemption exists to protect doctors from legal action in their treatment of patients. The attempts to circumvent Art. 52 have come from outside the European Patent Office, but also – if we take a literal reading of the EPC – from within the office itself.[50] If this is true, one must wonder why the exemptions were introduced into the EPC in the first place.

The grouping together of the exemptions together under Art. 52(2) has generally been seen to indicate that these are somehow conceptually related and that there is a logic behind the grouping which might allow us a context by which we could interpret these more accurately. Tapper, writing in 1983, did suggest that '[t]his scheme of exclusions apparently reveals a comprehensive intention to preclude the protection of ideas as such', but he – at that time – was also noting that it was 'not absolutely clear how much of the present [UK] law has been changed'.[51] Certainly there is some commonality between many of the elements – they are not at first sight of 'industrial application' and there is an abstract quality about each of these (at least to the 1970s engineer) which does lead to a sense that it is the abstractness (the lack of 'tangibility', to use Beresford's phrase[52]) which is common. Indeed, in T1173/97, the Board of Appeal suggested: '[t]he exclusion for patentability of programs for computers as such ... may be construed to mean that such programs are considered to be mere abstract creations, lacking in technical character.' But if there is no commonality between the group of abstract elements on the one hand and programs on the other, then we will get little help in interpretation of the program exemption. Pumfrey J has suggested[53] that there is actually little commonality between the items at all:

It is idle to pretend that it is easy to reconcile the different cases on these questions, but part of the difficulty, I think, is caused by an unspoken belief that the various

[48] *State Street* revisited this with a broad interpretation which may not – on a reading of the intentions of the 'founding fathers' – agree with their original intentions.
[49] See G1/83, G5/83, G6/83.
[50] Those arguing against software patents make recourse to Art. 52 and suggest that the Boards of Appeal have ignored its clear language.
[51] C. Tapper, *Computer Law* 3rd edn (London: Longman, 1983), p. 4.
[52] Beresford, *Patenting Software*.
[53] *Halliburton Energy Services, Inc.* v. *Smith International (North Sea) Ltd and others* [2005] EWHC 1623 (Pat), para. 212. Jacob LJ takes a similar position in *Aerotel*, discussed in Chapter 5.

excluded matters have something in common. In my view, they do not. They are a heterogeneous collection, some of which (aesthetic creations) have their own form of protection, others of which (discoveries, mathematical methods and scientific theories) have never been accepted as suitable subjects of monopolies on obvious, but different, policy grounds. The problems are really caused by (c) and (d), which, by reason of their exclusion 'only to the extent that the patent relates to such subject matter . . . as such' are remarkably difficult to assess in cases lying near the boundary, particularly as it is difficult to discern an underlying policy. For example, do we only exclude computer programs as such because computer programs as such are protected by copyright, like aesthetic creations which can likewise be used industrially? Or is there some other reason? Whatever the reason, surely it is not the same as the reason for excluding methods of doing business?

Pumfrey is focusing more upon the policy reasons as to why these exemptions are grouped together, rather than as to whether there is some other conceptual grounding for the drafting connection. It is not difficult to see what that grounding may have been – the fear that a system which was so close to pure mathematics and logic might have led to protection of 'inventions' which were far from those they believed it was intended to protect. However, given that the UK had the experience (from, for example, *Slee and Harris*[54]) of viewing software inventions as 'machine' such a perspective – that is, that the 'flood gates' would open if software inventions were allowed – is at odds with this UK experience.

There was also a further international element – apart from the PCT – in that sections of US industry were coming out against software patents. For example, IBM's customers were being harassed by holders of patents which were claimed to protect inventions residing in IBM software. IBM responded by lobbying:

> IBM addressed this problem by lobbying for software to be unpatentable. When the President set up a commission to study the U.S. patent system, a senior IBM manager was appointed head of this commission. Not surprisingly, the commission concluded that software should not be patentable because the patent examiners were not prepared to examine software patents.[55]

The President's Commission[56] had certainly been influential, focusing on the practical problems of implementing software examination, a prognosis followed by the Banks Committee in the UK. However, despite these concerns – and whatever it might have said in Art. 52, EPC – it is clear that applications involved computer-implemented inventions were

[54] [1966] RPC 194.

[55] Personal communication, Jim Sprowl. Sprowl was active as a patent attorney during this period and acted for clients who had purchased IBM bundled software and hardware.

[56] *President's Commission on the Patent System 'To promote the Useful Arts'* (1966) S. Doc. No. 5, 90th Congress, 1st Session (1967).

continuing to be forwarded to the USPTO and to the EPO for examination. For example, IBM despite its earlier aversion to protection to software patents sent an application within a few months of the EPO opening covering an idea the novelty of which – to my reading – resides very nearly in the notion of a business system ('Terminal, and transaction executing system using such a terminal'[57]) 'dressed up' as machine, and which was granted by the EPO in 1986.

In 1984 Gert Kolle, a member of DG5 and thus having some insight into what was happening in the EPO's examination system (DG2), was writing that:

> The chances of being granted a European patent or a national patent in a Continental European country are not good. Under the present state of the law it must be concluded that process and apparatus claims are not considered patentable if the difference from the prior art is constituted only by a novel or improved algorithm or is a related computer or organisational rule, or a computer program when applied to or carried out by conventional data processing equipment.[58]

Hanneman, too, at around the same time wrote[59] that, '[t]he existing possibilities for obtaining patent protection for computer-related inventions are very limited in Germany'. Clearly, Kolle and Hanneman were wrong and the very types of application they thought had little chance of success at the EPO and the Federal Patent Court were sitting in Den Haag or Munich awaiting successful prosecution. In the next section we outline why their predictions were errant.

The Boards of Appeal

What is striking about the patent system in Europe is that – despite it appearing to be chaotic and dispersed through various patent offices and national courts – it is actually a system which is highly centralised: that centralisation being around the Boards of Appeal of the EPO. There are several reasons for this: the move of patent applications from national

[57] EP0064592, filed January 1980: 'One such problem relates to the transmission of lengthy display messages from the host to the terminal during transaction execution. Because of this problem a number of predetermined standard messages must typically be stored in the terminal during initialization of the system, with the system thereafter relying on the standard messages stored in the terminal for communication with the consumer during execution of the transactions. In such systems the ability to communicate between the host and the consumer on [an ongoing], on-line basis and to compose messages custom-designed for a particular consumer or institution is lacking. Accordingly, it is an object of the invention to provide an improved transaction execution system.'

[58] Kolle, 'The Patentability of Computer Software', p. 61.

[59] H. W. Hanneman, *The Patentability of Computer Software: An International Guide to the Protection of Computer-related Inventions* (Deventer: Kluwer, 1985) p. 203.

offices to the EPO; the relative autonomy of the Boards of Appeal; national courts generally accepting the role of the boards;[60] and the impossibility of users appealing decisions to, for example, the Enlarged Board of Appeal. Note that there has been no appeal on computer-implemented inventions to the Enlarged Board and requests to refer the notion of 'technical effect' to the Enlarged Board have been refused by the boards of appeal.[61]

Also highly relevant is that the Boards of Appeal are given a workload which covers a specific technical area – which means that by various means[62] a collegiate approach can be taken to the processing of appeals. For computer-related inventions, the relevant board of appeal has primarily been 3.5.1. The board's first chance to view an appeal related to patentability rose in *Vicom*,[63] a case which is still central to European thinking on 'technical effect'. The *Vicom* application was related to a mathematical method of improving digital images, in particular by improving the speed of such processing. Images utilise large amounts of memory because each 'dot' (pixel) requires storage for its characteristics: colour and brightness in particular. A digital image which is 512 by 512 pixels would, the application suggested, require almost 60 million calculations as part of the enhancement processing. Clearly, a method of reducing this load substantially (they claimed a near four-fold increase) could offer potentially worthwhile time and hardware savings.[64] The application, filed just after the EPO began accepting work, was partly described as hardware (in the specification's Figs 1 and 2) but was primarily the application of a mathematical process. Following normal practice at that time, DG2 (the examination section of the EPO) refused it since:

(i) it related to a mathematical method which was not patentable by virtue of Art. 52(2) (a) and (3) EPC;

[60] An early example being R. Jacob, 'Industrial Property – Industry's Enemy', in *Intellectual Property Quarterly* 1: 3.

[61] Board 3.4.1 in T 0603/89 refused to refer a supposed contradiction between the Guidelines and a proposed BoA decision: 'However, this question does not need to be referred to the Enlarged Board of Appeal because the Board of Appeal hearing the present case considers itself able to answer it beyond any doubt by reference to the Convention; see also decision J 05/81, OJ EPO 1982, 155. For these reasons, the Board deems a decision of the Enlarged Board of Appeal as not necessary and the Appellant's auxiliary request is refused.'

[62] Anecdotally the author has heard that one member of Board 3.5.1 was kept away from appeals which may have related to 'technical character' in the 1980s due to his initial opposition to the concept. This may or may not be true.

[63] EP0005954, 'Method and apparatus for improved digital image processing', filed 1979, T0208/84.

[64] For example, a less powerful, less expensive machine could be used resulting in cost savings.

(ii) the dependent method claims 2, 4, 6, 7 did not add technical features as required by Rule 29(1) EPC; and
(iii) the apparatus claims 8–11 in the absence of supporting disclosure of novel apparatus were unacceptable in view of Article 52(1) and 54 EPC;

furthermore:

(iv) the Examining Division considered that the normal implementation of the claimed methods by a program run on a known computer could not be regarded as an invention in view of Art. 52(2)(c) and (3) EPC.

These criticisms, essentially, were correct: it *was* a mathematical method and despite the attempt to locate a hardware context there was a clear lack of the accepted 'technical features'. The applicant appealed this decision and this became the first of the computer-related inventions to reach the Boards of Appeal; moreover, since the examiners are required to follow the decisions of the Boards of Appeal,[65] this was their first opportunity to lay out practice for DG2.

There were three possible directions if they wished the appeal to be successful. First, the Board of Appeal could accept that this was a substantial improvement in applied mathematical technology and accept algorithms as patentable. Second, they could accept that this was a substantial development in applications software and allow protection as a program *per se*. Third, they could accept that the first and second alternatives were not feasible under Art. 52 and find an alternative solution. It was this third alternative which they took, essentially using a combination of the liberal UK approach set out in the UKPO note of 1969 – that is that there is a difference between a program *as such*, and a program when run on a machine – and that of the German federal patent court in *BCD-Conversion*. This third solution, the board suggested, was due to the *technical* nature of a program on a machine:

> A basic difference between a mathematical method and a technical process can be seen, however, in the fact that a mathematical method or a mathematical algorithm is carried out on numbers (whatever these numbers may represent) and provides a result also in numerical form, the mathematical method or algorithm being only an abstract concept prescribing how to operate on the numbers. No direct technical result is produced by the method as such. In contrast thereto, if a mathematical method is used in a technical process, that process is carried out on a physical entity (which may be a material object but equally an image stored as an

[65] Article 111(2), EPC: '. . . that department will be bound by the *ratio decidendi* of the Board of Appeal.'

electric signal) by some technical means implementing the method and provides as its result a certain change in that entity. The technical means might include a computer comprising suitable hardware or an appropriately programmed general purpose computer.

This reasoning – that what is important is the overall technical effect of an invention – has set the underlying grounds for EPO computer-related applications and remains European law. It is, though, hardly limpid in its clarity as many have complained. Indeed, when Van Den Berg – not a member of this *Vicom* board but a strong supporter of this technical effect reasoning in the Boards of Appeal when be became chair of 3.5.1 – attempted to explain[66] the logic behind the technical means framework, he provided the general framework of *Vicom* by suggesting:

Generally speaking, an invention which would be patentable according to conventional criteria should not be excluded from protection merely because modern technical means in the form of a computer program are used for its implementation.[67]

That is certainly a commendable objective, but it is not clear why any protection still had to be framed in the technical language of the prior technology. Van den Berg appeared to miss the contradiction which had been noted by Banks:

The board also pointed out, in addition to the reason already given, that it would seem illogical to grant protection for a technical process controlled by a suitably programmed computer but not for the computer itself when set up to execute the control routine.

To many it would seem illogical not to allow *protection for the software itself* when it is set upon on a suitably programmed computer to carry out a technical process controlled by a suitably programmed computer. It is a fiction to suppose that the novelty lies in anything other than the software. Van Den Berg – who was to be chairman of the board which eventually broached this contradiction and effectively acted as legislator for 'pure' software patents – pointed out that the 'boards of appeal cannot assume the role of legislator. They have to apply the law as it stands and cannot strive to meet wishes which are incompatible with the provisions of the European Patent Convention'.[68]

[66] Ironically in a text dedicated to the decisions of the Enlarged Board of Appeal.

[67] P. Van den Berg, 'Patentability of Computer-software-related Inventions', in Members of the Enlarged Board of Appeal of the EPO (Contributors) *The Law and Practice of the Enlarged Board of Appeal of the European Patent Office During Its First Ten Years* (Munich: Carl Heymans Verlag, 1996), p. 33.

[68] At p. 45.

The *Vicom* decision was clearly an attempt to locate what was an algorithmic process which was utilised in a program into a framework of protection which was not totally suitable. That is, it attempted to distinguish between an abstract concept and technical signals. A programmer – surely being the relevant person skilled in the art – would have difficulty in agreeing with such a highly artificial distinction. The programmer would not see the method as abstract at all – it could easily be implemented directly in a programming language; and he would certainly not be concerned about electrical signals – that would be handled by traditional input/output hardware. The core of the invention to the programmer is the processing of the data structure by means of the algorithm: there is nothing else in the invention of value.

To understand what was happening in the Boards of Appeal we have to consider their makeup. These are good (thus promoted) patent examiners with engineering expertise[69] who are concerned with looking to the heart of the application and rewarding inventive effort. The debate over late submitted evidence in oral hearings[70] was evidence of that: some boards were keen that, in order to give as good a decision on the technology as possible, they should be enabled to receive this, even if it put the other side at a disadvantage. This was based on the view that a hearing was not a court of law, but an investigation into an engineering solution to an engineering problem. Little wonder that such an approach would find it difficult to deny novelty and inventiveness to novel and inventive approaches. That Board 3.5.1 – in the view of even some members of other boards – were legislating rather than interpreting the EPC was something which could only be hidden by reference to an obscure and difficult-to-describe legal concept such as 'technical effect'. That it is hard to describe can be seen from the difficulty Van Den Berg had in explicating it in his article (and also the board in reasoning with it: 'the jurisprudence shows that the board's arguments vary from case to case'[71]) and that found by many other commentators in their elucidations. It is only when we stand back and, rather than using hermeneutics to try to understand 'technical effect', we have instead a less legalistic approach and see that the Boards of Appeal were using that same approach which the UK courts had done: integrating the new technologies within the patent system as a form of old technology rather than accept that it was

[69] Mainly, the legal role in any Board of Appeal is minor. See P. Leith, 'Judicial and Administrative Roles: the Patent Appellate System in a European Context', *Intellectual Property Quarterly* 1 (2001), 50–99. 'Engineering' is used in the widest sense.

[70] This was a debate internal to the BoA. See Leith, 'Judicial and Administrative Roles'.

[71] At p. 44.

actually a whole new and radical technology of its own. That is, they were describing a new type of machine with the terminology of an old one.

This approach was essentially that if we could imagine the invention as machine (apparatus or process as essential part of an apparatus), it was protectable; if not, it was not protectable. This holistic approach to examining the combined invention worked for many applications but for those where the invention was based in software, some manipulation of the way that this was described was required. This also meant that where the benefit of the invention was to a user of the program, it was – as Beresford argues of T89/0216 – necessary to remove the user from the picture completely:

> This case thus clearly demonstrates the importance of formulating the claims so as to specify clearly the structural and functional features of the programmed apparatus which embody the invention, as distinct from steps which are performed by an operator using the apparatus.[72]

It was not until the late 1990s that the some of the contradictions were finally dealt with by Van Den Berg with the IBM decisions (T97/0935 and T97/1173), particularly those which deal with the distinction between an invention which can be protected in software/hardware terms and one which can be protected as software alone. The two decisions were also used by the board as a means of outlining their interpretation of Art. 52 and its exemptions. We will look only at one of these.

T97/0935 dealt with an IBM application[73] to protect a 'Commit Procedure' which was claimed as a process. A commit procedure is very common in much programming, particularly of databases. A simple representation of the idea is that if Program A (say in an ATM 'cash machine') has collected information together on how much is to be paid out to the customer, date, time account number, etc. – and perhaps information from Program C as well – and this has to be saved in a database on a central computer running Program B. At some point, Program A will have to make the decision that it has the information and is ready to pass it all to B: it gives a 'Commit' command saying, basically, 'Go for it'. Usually everything works, but sometimes a fault arises and not all the information gets to B. This requires further processing to try to get the error sorted out and all the information to the required program – particularly if this is a sensitive procedure involving money and requiring security. IBM's invention was related to finding a way for Program B to

[72] Beresford, *Patenting Software*, p. 59.
[73] This was granted as EP0457112, 'Asynchronous resynchronization of a commit procedure'.

resolve the problem itself while letting Program A get on with other tasks (the next customer, for example) rather than waiting for confirmation.[74]

This is clearly a programming problem, residing in software – since it makes no difference which kinds of hardware it is run on. Commit procedures are, in fact, central to software such as SQL (a database system), which runs on a wide variety of forms of hardware. The hardware is certainly there, but the invention resides in the program if there is any invention at all.

The application contained a number of claims directed to the method of implementing the commit procedure (e.g. 'A method for resource recovery in a computer system running an application ...') which were held to be unproblematic by the examiner and also to be sufficiently novel and inventive to be awarded protection. The application, though, contained two further claims:

> 20. A computer program product directly loading in to the internal memory of a digital, computer, comprising software code portions for performing the steps of claim 1 when said product is run on a computer.
> 21. A computer program product stored on a computer usable medium ...

which clearly broke the machine metaphor that had been used to date: that is, that the protection was being requested for the software itself[75] rather than as part of a combined hardware/software machine. The examiner refused these claims and IBM appealed. The arguments used by IBM were wide ranging, including TRIPS,[76] examination practice at the Japanese and US offices and economic developments.[77] However, technical criteria are of prime interest here, where the examiner took the view that:

> Since the data medium and the program recorded thereon were not technically related, except for features which were already known for the prior art, the technical character of the computer program could not be derived from the physical character of the storage medium on which it was recorded. The technical character could not be derived from the method or system in which the computer was used.

[74] A process involving two 'equal' programs communicating in a highly ordered manner would be 'synchronous' whilst a process where communication is more random (that is, can be started by either and/or in stop-start mode or where one operates in subservient mode) is 'asynchronous'.

[75] Claim 20 during operation and claim 21 when stored on disk or other format.

[76] World Trade Organization, Agreement an Trade-Related Aspects of Intellectual Property Rights (TRIPS) 1994.

[77] The Board, clearly seeing itself as more than an administrative appeals facility, took upon itself to confirm its agreement with these arguments, yet stated the decision would be made on the interpretation of the EPC alone.

The examiner was thus following prior examination procedure – as outlined in the examination guidelines – and declined to provide protection for a software invention when claimed alone. The board's decision was, they decided, to be a review of the Art. 52 (in particular the meaning of 'as such'), rather than a review of whether the examiner had followed correct procedural practice. An outline of the logic which the board followed appears to be:

- programs are most usually mere abstract creations, lacking in technical character;
- programs which are abstract creations cannot be protected since they are programs 'as such';
- programs which have technical character are not abstract creations;
- programs which are not abstract creations (that is, have a technical character) are not programs 'as such';
- programs not 'as such' can be protected.

This is a reasoning here which a logician might consider to be circular and which takes us back, as the board suggested, to the main problem, which is the meaning of 'technical character'. It was at this point that the board changed the metaphor of machine to something closer to computing theory (that is, that software is itself 'machine-like') and suggested that a program itself can be technical, arguing that the prior case law of the boards supported this view: 'The case law thus allows an invention to be patentable when the basic idea underlying the invention resides in the computer program itself.'[78] The board then dropped the fiction that a patentable invention was in the machine which was part hardware and part software:

> It is self-evident that in this instance the basic idea underlying the invention resides in the computer program. It is also clear that, in such a case, the hardware on which the program is intended to run is outside the invention, ie the hardware is not part of the invention. It is the material object on which the physical changes carried out by running the program take place.[79]

It suggested that a view that protection could not be given for the underlying program was, the board held, illogical.[80] The appeal was allowed, the application sent back to the examination division and the guidelines were amended to incorporate this new practice.[81]

The practice of the examiner to ignore that it was the software which held the invention was illogical – as viewed by the programmer – but it

[78] Para. 7.4. [79] Para. 9.3. [80] Para. 9.8.
[81] Guidelines, Part C, Chapter IV, 2. Inventions.

was an illogical position into which the board had earlier put itself by requiring a model of a machine into which software must fit. In *Vicom* the board could have used exactly the same thinking to point out the lack of logic but did not: it gave protection, but only within the metaphor of traditional machine. The new decision certainly removed that one illogical factor, while trying to keep other 'abstract' inventions outwith protection – particularly 'algorithms' and 'business methods'. Unfortunately, as we shall see, allowing protection for software *per se* simply opens up other illogical positions which the insightful patent attorney can attack to benefit his client. For example, the Menashe patent discussed above shows how a patent attorney can persuade an examiner that the invention is a 'technical contribution'. The examiner had raised Art. 52(2) as a problem during examination[82] and the applicant's attorney replied that, by using the EPO's own problem and solution approach, the 'objective problem' could be stated as:

(a) How to limit the amount of data transmitted between the terminal and central computer in an interactive casino game, while at the same time ...
(b) Providing fair and tamper-proof play of a casino game outside the secure environs of a casino.

The objective problem so stated clearly does have the required technical nature required by the EPC.[83]

The examiner appears to have been persuaded that this was indeed a 'technical contribution' and the patent was granted, though someone more sceptical might read the patent as a 'business method patent', which we have been assured by various authorities is not permissible in Europe.

The missing element: the programmer's view

What may perhaps seem surprising is that the voice which was missing from this whole process of consideration of protection was that of the programmer. This should not be surprising, however from the earliest days of hardware, the software writer has been kept in the background and the written history of computing has, for example, been primarily the history of hardware.[84] Yet software developments have been an essential part of computing's success, even from those early days. As an example of

[82] Examination report, 6 November 1997, Application 94303526.1.
[83] Reply to examination report, 15 May 1998, Application 94303526.1.
[84] '... the programmer himself had a very modest view of his own work: his work derived all its significance from the existence of that wonderful machine. Because that was a unique machine, he knew only too well that his programs had only local significance and also,

that development, by the time of the inception of the European Patent Office many of the standard programming languages – including those based upon object-oriented programming – had already been produced.[85] These languages developed from Fortran – an artefact which John Backus has suggested many of those knowledgeable in the state of the art at the time felt was impossible to produce. And that, too, had been based upon the original developments of others – for example, Grace Hopper, who produced the first compiler: 'I had a running compiler', said Grace, 'and nobody would touch it because, they carefully told me, computers could only do arithmetic; they could not do programs. It was a selling job to get people to try it. I think with any new idea, because people are allergic to change, you have to get out and sell the idea.'[86] The inventive radicalism of this step can be compared with the move from the Newcomen Engine to that of Watt's separate condenser, but whilst Watt's patent lasted for some 25 years and – some argue – held up the development of the steam engine,[87] Hopper's work was not protected via the patent system and was open to all to develop and improve upon – as proponents of the open source model frequently remind us is the advantage of a patent-free technology.

Despite the existence from an early date of this culture and expertise in programming, when we look at patent applications and the process of examination, the view which is usually missing is that of the programmer. It is as though the programmer's view of technology were considered irrelevant. This is true – it was not relevant. In the attempt to fit the new computing technology into an appropriate and patentable classification, his voice was ignored and the model which was used was that of the classical 'machine'. 'Technical effect' is essentially a dynamic concept

because it was patently obvious that this machine would have a limited lifetime, he knew that very little of his work would have a lasting value.' Dijkstra, 'The Humble Programmer', 859–66.

[85] Fortran in 1954, Ada in 1979. See 'The Programming Languages Genealogy Project', at http://www.everything2.com.

[86] http://www.thetech.org. Hopper also worked on the development of Flowmatic, which became Cobol, the business programming language. Note that, later, it was seen that the software notion of a compiler could also be used to produce hardware – see K. Keutzer and W. Wolf, 'Anatomy of a Hardware Compiler', Proceedings of the SIGPLAN '88 conference on Programming Language Design and Implementation (1988) pp. 95–104, in which they claimed to have used to design chips with 30–60,000 transistors.

[87] The argument is that Watt's supremacy in the field (because he controlled the technology) led to a lack of research and development so that even after the patents expired John Farey, an early patent agent (1827), could write: 'Men of superior intellect, who might have been induced to investigate the subject, have been led to suppose that nothing further remains to be perfected . . .' See H. I. Dutton, *The Patent System and Inventive Activity During the Industrial Revolution, 1750–1852* (Manchester: Manchester University Press, 1984) for a more rounded picture of Watt, Farey and early patent activity.

which mirrors the definition of 'machine' suggested by Reuleaux: 'A machine is a combination of solid bodies, so arranged as to compel the mechanical forces of nature to perform work as a result of certain determinate movements.'[88] The legal requirement in European patentability has thus been looking for a change of physical state, which usually accompanies work effected. This is not a programmer's conception of the machine: work is not the output of the new technology, but rather it is the processing of *information*, a procedure which involves no Reuleaux-like work performance except moving the contents of one data structure to another, as occurs in IBM's asynchronous commit procedure.

To the programmer there was a different kind of machine which was highly relevant and had dominated their world since many programmers – after the 1950s and the work of Hopper *et al.* – had successfully moved away from consideration of the machine *as such*, to consideration of the machine as *tangible* yet *virtual*. Thus the development of operating systems had advanced to the point where 'virtual machines' were the order of the day – a self-contained operating environment behaving as if it were a separate computer – which (despite the nature of the underlying implementations) allowed the programmer to see the software as a machine into which he could insert data processing tasks. For example, the *Virtual Machine* – an IBM mainframe operating system – was originally developed by its customers and eventually adopted as an IBM system product (VM/SP) running multiple operating systems (OS) within the computer at the same time, each one running its own programs. A virtual machine is the *construct of a program that behaves like a real machine*, so that an OS, or other program written to run alone on a real machine, runs as though on the hardware itself:

> Creasy and Comeau spent the last week of 1964 joyfully brainstorming the design of CP-40, a new kind of operating system, a system that would provide not only virtual memory, but also virtual *machines*. They had seen that the cleanest way to protect users from one another (and to preserve compatibility as the new System/360 design evolved) was to use the System/360 *Principles of Operations* manual to describe the user's interface to the Control Program. Each user would have a complete System/360 virtual machine (which at first was called a 'pseudo-machine').[89]

[88] F. Reuleaux, *The Kinematics of Machinery: Outline of a Theory of Machines* (1876) reprinted (New York: Dover Publications, 1963) p. 35, available online at http://historical.library.cornell.edu.

[89] M. Varian, 'VM and the VM Community: Past, Present, and Future' (1997), available online at http://www.princeton.edu/~melinda/25paper.pdf (see p. 10).

To the programmer working on the design of a data object or a procedure, the task is tangible: the programmer views the objects he is dealing with as physical entities – we see this clearly in the use of diagrams by computer scientists when they explain what they are trying to do (particularly when they communicate with each other). The programmer is building a machine with the same mode of thinking as the eighteenth-century millwright John Rennie placing iron cogs and drive wheels in relationship to the power source at the Albion Mill[90]: that is, as a physical and tangible system – even though the actuality is non-physical and intangible. Where Rennie was concerned with reducing friction and maximising power use the programmer – especially in the early days – was concerned with reducing memory usage and maximising throughput to the processor. The thinking was the same, but the nature of the machine differed substantially – one tangible, one virtual.

The histories of programming which are now beginning to appear validate the view that real inventive advances can be found in software *as such*, rather than in some amorphous 'machine' of the patent examiner. See, for example, Per Brinch Hansen's history of concurrent programming,[91] which emphasises invention at the conceptual (i.e. virtual) level. Concurrent hardware is easy to build (simply attaching a number of processors in parallel, for example) but what remains difficult is the software control of this hardware, and that the inventive element – if it exists – lies with the software researcher and developer rather than the hardware engineer. Reading these histories we see that – just as in development in the sciences – terminology and definition of concepts are at the heart of invention.

It was this situation – where software was being seen as an immensely powerful and malleable system – which the patent system was attempting to control. Basically, a new form of technology had developed and despite the rhetoric of the patent system that its goal was reward for new technological development, the system was only prepared to accept this new technology on the basis of nineteenth-century notions of machine, not in the new manner in which the programmer saw this new machine. Such legal fictions led to contradictions at the heart of the patent examination system – to protect one artefact by basing it upon the metaphor of a different artefact must lead to problems – which are difficult to

[90] See S. Smiles, *Lives of the Engineers, with an Account of their Principal Works: Comprising Also a History of Inland Communication in Britain* (London: J. Murray, 1861–2).

[91] P. B. Hansen, 'The Invention of Concurrent Programming', in P. B. Hansen, (ed.), *The Origin of Concurrent Programming: From Semaphores to Remote Procedure Calls* (New York: Springer-Verlag, 2002), pp. 3–61.

uphold and thus we had *Vicom* and the later Board of Appeal decisions. But these later decisions too have contradictions at their core and we must wonder whether they can continue to hold against the pressures desiring protection for the various kinds of technologies based upon this new rampant model of virtual machine upon which has developed a new technology.

2 Software as software

The time will come when manufacturers will give away computers so as to be able to sell software.[1]

Introduction

In the preceding chapter we looked in overview at how software had been viewed by the patent system prior to the IBM decisions (T 1173/97 and T 935/97) and noted that the system's emphasis upon integrating 'computer-related inventions' within the framework of a tangible, physical machine ignored the perspective which the programmer has of this new technology. This is not to say that there is one single programmer-centric perspective which we might simply persuade those in the patent system to adopt in their discussion of technology: far from it. However, we can paint a reasonably accurate picture of how programmers view their technology and note where there is a divergence between the technical and the legal.

The point of painting this picture is that it should allow us to mould our legal perspective upon a more truthful model of software than that of a 'soft' hardware device. Lawyers have – Shklar suggested – a legalistic view which encourages the belief that legal reasoning can meet and overcome the challenges set by all other disciplines:[2] perhaps this is so, but there is

[1] H. Aiken, reported in I.B. Cohen, *Howard Aiken: Portrait of a Computer Pioneer* (Cambridge, MA: MIT Press, 1973), p. 244. We are not there yet – but OS and 'office' software can now cost more than the hardware upon which it runs. There are also instances where software is given away free and profit is made through provision of services – for example, in the open source 'marketplace' – which demonstrates further that hardware is becoming of lesser value to the market than it was in the period when the EPC was being constructed.

[2] J.N. Shklar, *Legalism – Law, Morals and Political Trials* (Cambridge, MA: Harvard University Press, 1964): 'Law is endowed with its own discrete, integral history, its own "science," and its own values, which are all treated as a single "block" sealed off from general social history, from general social theory, from politics, and from morality ... it aims at preserving law from irrelevant considerations ...'

anecdotal evidence that these other fields are not always persuaded of the power of purely legalistic approaches. In a research interview, one London IP lawyer rued the fact that she spent considerable time preparing cases for her IT clients only to find that, on their first appearance before the judge, the latter's lack of IT knowledge caused such anxiety in the parties that they quickly came to agreement or headed off to arbitration. These were not patent cases but in many European countries patent hearings will be held in front of non-expert judges – little wonder that one of the fears of the Community Patent has been that of forum shopping by litigants.[3] The best law is that where irrelevant considerations are held at bay, but relevant ones are taken seriously – and in the field of patentability, the nature of software is a relevant issue every bit as much as the nature of DNA is in genetic patenting.

The question of the relationship of law and science is something which has become of interest to researchers over the past decade or so, and writers such as Jasanoff[4] have pointed to the difficulties found when two systems which claim authoritative roles – law and science – meet in the courtroom. This confluence is difficult enough in Europe, where it is the norm to hear debates about inventions in front of judges, but in the US the position appears – to this author – to be very much worse, with an expectation that a jury of peers, perhaps with limited contact with science and technology in their education and little understanding of how the many basic electronic tools operate (the mobile phone, for instance), will be able to decide upon the claims made in advanced technology disputes.[5] Those countries which utilise specialist judges in validity matters (UK and Germany, for example) may have an advantage over other approaches, but there are still problematical areas, for example over the role of expert witnesses when the field discussed is outwith the direct

[3] 'Many witnesses could not accept the risk that a patent could be invalidated by any national court. Their concern is that the judge might not have sufficient knowledge and experience of patents'. *The Community Patent and the Patent System in Europe, Select Committee on the European Communities*, House of Lords, Session 1997–98, 26th Report, HL Paper 115, para. 85.

[4] S. Janasoff, *Science at the Bar: Law, Science and Technology in America* (Cambridge, MA: Harvard University Press, 1995).

[5] 'Honest to God, I don't see how you could try a patent matter to a jury. Goodness, I've gotten involved in a few of these things. It's like somebody hit you between your eyes with a four–by–four. It's factually so complicated.' Quoted by K. A. Moore, 'Judges, Juries, and Patent Cases – An Empirical Peek Inside the Black Box', *98 Mich. L. Rev. (2000), 365*. A. B. Jaffe and J. Lerner, *Innovation and its Discontents: How Our Broken Patent System is Endangering Innovation and Progress, and What to Do About It* (Princeton: Princeton University Press, 2004), are also highly critical of the role of juries in the US system. They blame the CAFC.

expertise of the judge.[6] In research interviews in London this author was told of dashes by litigators to get their hands on the most prestigious expert witnesses before the other side got them. It was best, I was told, if they had won a Nobel Prize. The witnesses – academics – appeared more than happy to drop whatever they were doing and become well-paid proponents for parties who paid the piper, leaving the judge with the task of analysing the value of their scientific contribution to the litigation. The authors of *Terrell* point to the complaints of patentees that the witnesses are considerably over-skilled given that the reader of a patent is someone generally skilled in the art, rather than being at the very top of the scientific hierarchy.[7] The recent procedural rules in the Patents Court in London can be seen as an attempt, in part, to mitigate such problems: a skeleton argument document[8] must be prepared and agreed by the parties in which, in effect, they point to areas where their respective experts disagree. A different and decentralising approach is one where expert judges might be utilised, for example, drawing them from the Boards of Appeal of the EPO,[9] or – using a more centralised approach – setting up a distinct European Patent Court with expert judges as promoted by the EPC member state Working Group on Litigation,[10] which has proposed a draft statute that includes no mention of expert witnesses.

Even for courts such as those in London, where there exist judges with patent expertise and access to a long-established patent office, expert witness input to litigation can be problematic. In Halliburton's application[11] to have their patents re-registered, the court noted the

[6] See also *Cantor Fitzgerald International* v. *Tradition (UK) Ltd and others* [2001] EWCA Civ 942, with discussion of a problem caused by an expert witness in a software copyright case (see para. 70): 'But I was left with the strong feeling that Mr Wise was acting as an advocate.' *Per* Pumfrey J.

[7] S. Thorley, R. Miller, G. Burkill and C. Birss, *Terrell on the Law of Patents* 15th edn (London: Sweet & Maxwell, 2000). The authors state (at para. 6.25): 'However, the courts have generally found it very helpful to have the best possible guidance from persons steeped in the art, and have been able to make allowance where necessary.'

[8] Para. 11 of *The Patents Court Guide*, Issued 12 November 2003, states: 'Further the parties should endeavour to produce a composite document setting forth the matters alleged to form part of the common general knowledge and, where they disagree, what that disagreement is.' Also see paras. 12, 13 and 14. Available at Court Service website: http://www.hmcourts-service.gov.uk/infoabout/patents/crt_guide.htm

[9] P. Leith, 'Revision of the EPC, the Community Patent Regulation and "European Technical judges"', *European Intellectual Property Review* 23(5) (2001), 250–4.

[10] Working Party on Litigation, *Draft Statute of the European Patent Court* (16 February 2004), available at the EPO website:

[11] *Halliburton Energy Services Inc* v. *Smith International (North Sea) Ltd and others* [2006] EWCA Civ 185. The practice was further clarified at appeal. See *Halliburton Energy Services Inc* v. *Smith International (North Sea) Ltd and others* [2006] EWCA Civ 1715, where it was stated (at para. 7) that: 'We envisage that in the future, where a scientific

problem of finding – given that only one party would be represented – suitable expertise to provide some opposition to the patentee.[12]

In computing, the field has become so broad that it is impossible for a computer scientist fully to appreciate what is happening in all the other sub-fields: how can someone involved in database query language design understand the constraints under which compiler designers operate, and how can either get to grips with the use of formal definition methods in program verification? Clearly they can, but only by following a process similar to that of a contract lawyer who wishes to become an expert in computer crime: that is, by becoming an expert in the required field itself. Having said that, there is a commonality between all these different areas – that is, the task of programming. All sub-areas of computing require expertise in programming as the basis of their other expertise. Programming is thus a core element in understanding computing and, if we wish to take software seriously in the courtroom, we need to have an understanding of the programming process.[13]

In Chapter 5 I will look more closely at the particular nature of algorithms with respect to legal interpretation, but here simply use the notion of 'a sequence of steps' which is translated into a textual program.

What underlies programming?

A programmer always works with a virtual environment – that is, a 'world' moulded either by himself or by others. It is easiest to see this world as a model constructed in the mind and transferred over to code. As Perlis suggested:

adviser is appointed, a similar record of the part he or she has played will normally be provided to the parties with, of course, also an opportunity being given to them to comment on it.'

[12] Jacob LJ: 'That is whether or not this court should have the assistance of a scientific advisor. [Halliburton] said, undoubtedly. In those circumstances, it obviously makes sense to direct there should be such an adviser in principle. [Halliburton] suggested that one need not go to the level of a Fellow of the Royal Society or the like, but maybe assistance could be provided by a patent office examiner if the Comptroller is not going to appear. The patent office does not know whether they could provide any such person. What is clear is that if any person is provided, whether from the patent office or elsewhere, is that the costs of that provision would also fall upon the patentees.' In the event, Professor David Limebeer, Head of the Department of Electrical and Electronic Engineering and Professor of Control Engineering at Imperial College London, was appointed.

[13] From the late 1980s until 2004 the author ran an LLM in Computers and Law where students were required to undertake quite substantial programming tasks as part of the degree. This confirmed my conviction that technical understanding is essential to understand law and technology.

Every computer program is a model, hatched in the mind, of a real or mental process. These processes, arising from human experience and thought, are huge in number, intricate in detail, and at any time only partially understood. They are modelled to our permanent satisfaction rarely by our computer programs.[14]

Such an insightful description will immediately strike a chord of recognition in the programmer, but it is unlikely to do so in the mind of the lawyer. Given that the latter could fail to pick up the subtleties which Perlis has compressed into this paragraph, we need to decompress his description and make it accessible. We will do so by utilising a thought experiment based upon early computing technology in the context of current law. The advantage of this is that we have a much-simplified view of the programming process (but one which was essentially historically accurate and comprises the core which is still present in programming) which lets us more easily examine the link between programming and patent and other legal protections. However, we need to remind ourselves that this example problem is simply a useful way to understand the nature of programming. Few programs written today are in any way related in size to our example – quite the reverse. A major problem for computing now is how to manage complexity caused by size of program, a problem which was evident even as early as the late 1960s and 1970s. Dijkstra clearly linked it to the advances in hardware, but it is a problem of complexity which has now outgrown that of hardware, the latter being more readily conceptualised. The inventor of the solution to the 'software crisis' – the name given in the 1970s to the failure to be able to produce programs on time, to cost, and which do what they are intended to do – should surely be rewarded:

[T]he major cause [of the software crisis] is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.[15]

The problem

The technical problem is that of manipulation of objects in a hazardous environment – the first nuclear power stations in the late 1950s used

[14] Foreword to H. Abelson, G. J. Sussman and J. Sussman, *Structure and Interpretation of Computer Programs* (Cambridge, MA: MIT Press, 1985).

[15] E. W. Dijkstra, 'ACM Turing Lecture: The Humble Programmer', *Commun. ACM* 15(10) (1972), 859–66.

trained monkeys[16] to manipulate the different shapes of nuclear materials within the maximum radiation zone. These materials were prepared in the shape of pyramids, cubes and balls. The monkeys were trained to understand that only certain objects could be placed on top of other objects: balls can only be placed upon the table (i.e. they cannot be placed upon cubes or pyramids) and cubes cannot be placed upon pyramids. Nothing can be put on top of balls. Blocks can be stacked to a maximum of five high.[17] The monkeys have been effective but, unfortunately, do not appear to live long and the training expenses are continual. Also, there are some worries from the nuclear health and safety inspectorate that not all monkeys may be able reliably to count above four. Newly invented mechanical robotic devices are being seen as replacements for the monkeys but their mechanical nature means that they require control from outwith the radiation zone. A computer program, we have radically suggested, could be utilised to control the newly invented forms of robotic arm (Fig. 2.1) and also ensure correct material handling. Further, we tell them that, by using a computer, the various moves taken can be stored on external media for future reference and analysis.

Possible solutions

A programmer – on meeting this problem – would immediately begin to consider a number of options. In classical programming,[18] where the definition of a program is:

Program = Algorithm + Data Representation

the programmer would probably think first about how the *data representation* (or 'data structure') of the problem space could be *modelled*. That is, how a computer-oriented representation of the information to be stored and manipulated could be arranged. The non-programmer would expect discussion of the algorithm to be primary, since this is the aspect which is mainly discussed, particularly in patent matters. However, for many programs the actual algorithm relates to well-known steps which are

[16] To the best of the author's knowledge, monkeys were never actually used in such a role.

[17] A computer scientist would recognise the 'blocks world' problem used by early AI researchers. See J. Slaney and S. Thiebaux, 'Blocks World revisited', in *Artificial Intelligence* 125 (2001), 119–53 for a recent discussion. This author has also used it in teaching formal methods. See P. Leith, 'A Programmed, Skeleton Formal Specification Method:, The OUFDM' *Computer Journal* 30 (4) (1987), 337–42.

[18] Usually referred to as 'procedural programming' and most easily described by flowcharts to represent the steps in the procedures.

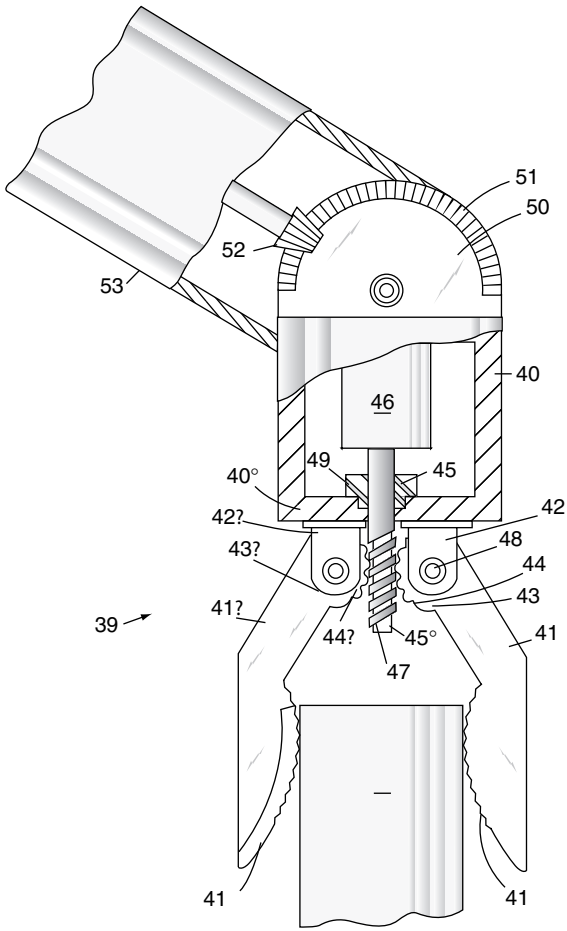

Fig. 2.1. US202449 Article Manipulation Device (filed 1963)

to be taken, and involves moving, comparing and sometimes doing calculations on the data representation. Thus the first task in programming is usually to design the data representation which best mirrors the problem in the real world. There are many possible representational solutions. For example, the solution could be modelled as a data table with locations being used to represent positions on the nuclear table and blocks stored on it, as in Fig. 2.2.

Of course, we could also do essentially the same thing in 'hardware' terms and would need just a few bits for each entry and use binary representations for each entity with each table position being occupied

Table Position 1	Ball			
Table Position 2	Cube	Ball		
Table Position 3	Cube	Cube	Cube	Pyramid
Table Position 4	Cube	Ball		
"				
"				

Fig. 2.2. A table-based data structure

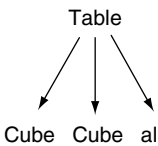

Fig. 2.3. A tree-based data structure

by sixteen bits.[19] If we then give the cube the value of 1, the cylinder 2, and the ball 3, then the first memory location might contain the value 11223 (in binary – 10101111010111). Thus, for each location we use a digit to represent the nuclear materials, and we know that a location which holds 11223 has two cubes, two cylinders and a ball in the column.

Alternatively, we could use a tree structure, which is a slightly more abstract model but can be implemented so that it is actually held in the hardware format just outlined, although we think of it as being a different model as in Fig. 2.3.

There are other ways in which the programmer could conceive of handling this information. Given that our problem is located in the early 1960s, we could store this information outwith the machine on paper cards or tape and run these back through the machine when required, but that would involve more user actions than we might want. Other, non-classical, programming techniques, such as object-oriented programming (OOP), could also be utilised, but since techniques such as OOP were developed later in the 1960s, we shall not discuss these at present.

For each of the different data representations, we can use a relevant algorithm for manipulation. Thus our basic algorithm for the table might be:

1. Accept user input.
2. Test whether there are free spaces in table at required position.

[19] Computer organisation is based upon multiples of two due to its being a 'binary' machine.

3. Test whether object in position allows insertion.
4. If test in steps 2 and 3 ok, then insert at position.

The tree structure algorithm, on the other hand, would involve moving over the tree to find out what the other 'branches' contained.[20] In effect, we are performing the same task with each of the algorithms but – as expressed – each algorithm is different, in that one is manipulating a table, one a tree and one a computer word. Since each one reflects a different model, it will be expressed in textual or diagrammatic terms as different from the other algorithms.

What are we to make of this? Exactly the same information is being processed (where objects have been placed) but there are a variety of different *models* being used by the programmer to solve the problem of how to represent the data. It is important to realise that these models become reality for the programmer – he moves from setting up the virtual objects to their being the world in which he sites himself. When we move from simple block world programs to more complicated programs, the complexity of these virtual worlds with virtual data representations becomes a significant problem to the programmer. Why? Primarily, because the virtual representations are representations of a more complex world and these are – as Perlis pointed out – often worlds which are only partially understood by the programmer. Also, a program which is produced by one programmer and his own personal visual representation becomes a qualitatively different object when it involves a team of several hundred who are located in various parts of the world and – in terms of program maintenance – in time measured over decades.[21]

The programmer handles this complexity by:

1. arranging a specification of the problem – the more complex the problem, the more formal the specification should be, some suggest;
2. producing a solution in terms of a virtual world which matches this specification;
3. using programming tools which are designed work with the objects in this virtual world (e.g. Cobol for business programming).

The success of this methodology is not guaranteed. For example, in business applications of the computer, the systems analyst's role is to produce a specification, but since this is carried out through document analysis and interview, it may miss out on important (tacit) information about the processing and why certain tasks are done in certain ways. This

[20] Tree traversal is one of the fundamental tasks in computing.

[21] The 'Year 2000 problem' was due to much code still containing earlier pieces of program which had been written when memory was scarce and saving a few bytes was a valuable programming technique.

faulty specification then becomes the basis of the solution. In the 1960s many business applications were simple billing-type problems and related to the manipulation of accounting records, so that the programming tool used was mainly Cobol, a programming language where the virtual objects were 'files', 'records' etc. But as the machines have become more powerful (allowing ever-larger programs), the programs too have become more complex and the difficulty of specifying problems has risen substantially.

In simple terms what we have are two parallel worlds: the real world in which the eventual program is to be located and the virtual world of the programmer. As Perlis suggests, sometimes, these two worlds do not totally meet.[22] What is important is to realise that one programmer's virtual model, which implements exactly the same model as another's, may appear to be very different indeed if examined via its internal structure. This is part of the problem inherent in software patent examinations, where a software description will differ from that of an engineering artefact such as the robot hand above, the latter lacking the conceptual malleability of programming.

Our user input will be a formal program, but one which is interpreted rather than compiled – that is, the user types in one line of the desired program at a time and this is executed immediately by our interpreting program.[23] In simple problems such as nuclear materials handling this is a sufficiently effective procedure and uses the same interpretative technology as Lisp, a programming language designed in 1958.[24] The formal definition[25] of our language is laid out in Fig. 2.4.

The user will input (using the DATA statement) to the terminal information first about what blocks are available and their names:

```
DATA BLOCK1 CUBE
DATA BLOCK2 BALL
```

[22] Linus Torvalds – innovator of Linux – has suggested, 'A "spec" is close to useless. I have *never* seen a spec that was both big enough to be useful *and* accurate. And I have seen *lots* of total crap work that was based on specs. It's *the* single worst way to write software, because it by definition means that the software was written to match theory, not reality.' Published email, September 2005. See http://www.kerneltrap.org.

[23] A compiler translates a complete program into machine code before execution, an interpreter translates into machine code one line at a time. BASIC is an example of a language which allows both. The model of interpreter is very useful in debugging where single steps can be taken to find out where bugs lie.

[24] Though given its novelty and inventiveness, 'designed' may not be the best description. See J. McCarthy, *History of Lisp* (Artificial Intelligence Laboratory, Stanford University 1996), available online at http://portal.acm.org/citation.cfm?id=808387&dl=ACM&coll=portal

[25] This definition sets out the syntax – but not the semantics of the language – using BNF, a standard description method for languages used first to define Algol 60. There is some debate over whether the 1959 'inventor' was John Backus (thus Backus Normal format) or whether Peter Naur was involved (Backus Naur Format).

```
<PROGRAM>                 ::=<DATABLOCK>¬ CLEAR¬
                             <INSTRUCTIONBLOCK>
<DATABLOCK>               ::=<BLOCKDESCRIPTIONI>
                             <BLOCKDESCRIPTION>¬
<DATABLOCK>
<BLOCKDESCRIPTION>        ::= DATA¬ <BLOCKNAME>¬ <TYPE>
<BLOCKNAME>               ::= BLOCK+<DIGIT>
<TYPE>                    ::=BALL¬ CUBE¬ PYRAMID
<INSTRUCTIONBLOCK>        ::=<INSTRUCTION> | <INSTRUCTION>¬
                             <INSTRUCTIONBLOCK> |
<INSTRUCTION>             ::= CLEAR | SAVEMOVES | <PLACEINSTRUCTION> |
                             <PUTONINSTRUCTION> |
                             <TAKEOFFINSTRUCTION> | <LOOPINSTRUCTION>
<PLACEINSTRUCTION>        ::= PLACE¬ <BLOCKNAME>
<PUTONINSTRUCTION>        ::=PUT¬ <BLOCKNAME>¬ ON¬ <BLOCKNAME>
<TAKEOFFINSTRUCTION>      ::=TAKE¬ <BLOCKNAME>¬ FORM¬ <BLOCKNAME>
<LOOPINSTRUCTION>         ::= LOOP¬ <DIGIT>¬ TIMES¬
                             <INSTRUCTIONBLOCK>¬ REPEAT
<DIGIT>                   ::=1 1 2 1 3 1 4 1 5
```

Fig. 2.4. The Nuclear Blocks Handling Language

If this is syntactically correct, then our interpreter will accept it and further statements about blocks to be accepted. CLEAR is input to tell the program that we are about to start manipulation (or to remove all blocks at once). PLACE, PUT and TAKE are used to manipulate the blocks:

```
PLACE BLOCK1
PUT BLOCK2 ON BLOCK1
TAKE BLOCK2 FROM BLOCK1
```

The interpreter will return an error message to the user in this example, since it tests that a ball cannot be placed on a cube and refuses to execute the statement. The LOOP/REPEAT statement is useful to save retyping if we want to build several blocks upon each other. SAVEMOVES is a command which will write the moves made to a backup storage device (tape perhaps) thus becoming part of a database for future reference.

Where is the algorithm?

To read some legally oriented articles on software patents, one might get the feeling that the algorithm is the most important element in software and the only element which might require protection.[26] That is not true,

[26] D. S. Chisum, 'The Patentability of Algorithms', 47 *U. Pitt L. Rev.* (1986), 959, for example, does not put algorithms into any real programming context.

since in many programs the algorithm follows from the choice of data representation. Of course, we could write a program and do analysis of how it performs which will help us improve the algorithm if necessary (and perhaps change the data representation) so that bottlenecks in processing can be speeded up. One example of this would be the 'invention' of the 'single pass' compiler which runs over program code changing it into machine code in, as the name suggests, one pass[27] rather than the earlier two or three passes. An example closer to the lawyer of the interlinked nature of algorithm and data structure is the way that legal texts are held in systems such as Lexis, Westlaw or Bailii: here each document is processed and each relevant word is indexed as to location.[28] We can apply many different algorithms to the same data structure but each is bound to the particular way that we initially chose to index words.

Our example above reflects normal programming: that is, that the algorithm and data representation are intrinsically linked together. For example, the algorithm which we will use to manipulate the blocks depends upon how we represent these blocks. If we have a patent system where algorithms are protectable, surely we should have protection for data representations too? And, conversely, if we can protect data structures, then surely algorithms must likewise be protected? And if both of these can be protected, can we go one step further and protect the mental models (the 'ideas') upon which they are based?

Communicating the virtual worlds – the ideogram

The 'article-manipulation device' (above) shows the basic elements of engineering description by using visual representation. We can see the form of the object; can basically work out how it fits together and see that, while we may not be able to tell details of the thread forms, gearing and motor power or rpm required, we can get sufficient information to understand the idea at a glance. This is important, since it is usually the idea which is being protected by the patent system rather than the specific manufacturing tolerances (though aspects of that, too, may be developed to become 'inventive'). Engineering drawing itself rather than that used in patent specifications is a highly organised and developed process with its

[27] Changing the algorithm here certainly speeds up the initial production of machine code, but does it by producing less efficient code and requires a language to be designed in such a way (i.e. with suitable data representations) allow this. It also requires larger memory resources than was available in the earliest machines.

[28] See P. Leith and A. Hoey, *The Computerised Lawyer*, 2nd edn (London: Springer-Verlag, 1998).

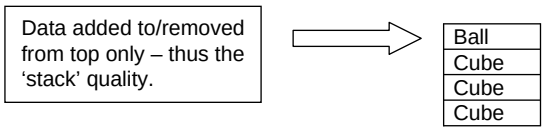

Fig. 2.5. A stack data structure

own visual grammar and standards[29] and will detail the relevant information relating to tolerances, materials etc. The same is true of chemical descriptions where there is a visual language which describes the structure and/or nature of processes and compounds (this can either be a diagram or a word which represents the structure of the compound). For example, Roy Plunkett's Teflon patent (US 2,230,654, 1941) was able to utilise the single word 'Tetrafluoroethylene' to describe exactly the compound which he had been able to polymerise.[30] Electrical and electronic specifications utilise a commonly accepted grammar which both indicates well-known outline functions ('op amp', say) and specific implementations. Generally, the situation with regard to the visual depictions of the main technical fields supported by the patent system is that there are well-accepted and well-understood ways of communicating the underlying ideas. These means help reduce possible misinterpretation of the application and, if granted, also make the specification less liable to misreading.

This, unfortunately, is not the case with computing – there is little agreed language or visual form of representation to describe the virtual worlds which the programmer builds. In large part this has been due to the freedom which programmers have had in developing their ideas with no central authority (or 'institute', to use the Victorian concept) which has moulded the terminology to date. For example, even a very simple programming concept – which is usually now referred to as 'a stack' because items are added to it on the top and taken off only from the top again (as in Fig. 2.5) – has had a chequered history of naming.

Donald Knuth – in his attempt to bring some formality to computer science – noted that:

> Many people who realized the importance of stacks and queues independently have given other names to these structures: stacks have been called push-down lists, reversion storages, cellars, nesting stores, piles, last-in-first-out ('LIFO') lists, and even yo-yo lists![31]

[29] For example, BS 8888 is a British drawing standard which includes European standards.
[30] Claim 1 – in total – simply referred to 'Polymerized tetraflouroethylene'.
[31] D. Knuth, *Fundamental Algorithms: The Art of Computer Programming* 2nd edn, vol. 1 (Reading, MA: Addison-Wesley, 1973), p. 236.

Knuth argued that these were errantly called 'push-down' and 'pop-up' stacks or lists because, though convenient for brevity and analogy, they 'falsely imply a motion of the whole list within computer memory'[32] but the terminology remains in use.[33] The related data structure, the queue (which mirrors a stack but where data is put onto the top of the stack and removed from the bottom) was also noted by Knuth to have been called a circular store or first-in-first-out ('FIFO') list, descriptions which have still not totally withered away.[34] The objection that Knuth has to this description may have been that he was keen that programmers had an understanding of the hardware underlying any program: in his multi-volume *Art of Computer Programming*, he specifically chose a language for his examples which reflected machine level programming, since: 'a person who is more than casually interested in computers should be well schooled in machine language, since it is a fundamental part of the computer'.[35] In machine language terms, there is of course no such thing as a stack which looks like a pile of dishes awaiting cleaning:[36] there is a data location with one associated list pointer (another data location) which points to the next data location on the list. A further single pointer/data location points to the last data location of the stack. Many programmers today – I suspect – would have little real understanding of what is happening at the machine level, either in hardware or in terms of basic software processes, since they are working at a higher level of abstraction; that is, with a different virtual model from Knuth. The ubiquity of the stack model has meant that it has been included at the abstract level in programming languages so that programmers can work with it without having to be concerned with the implementation details (and that implementational virtual model). Further, some languages such as Forth and PostScript are entirely based upon a stack model.[37]

Computing definitions continue to be problematical and the field lacks a coherent and respectable dictionary of terms. The US Patent Office, worried by the fact that Wikipedia.org was editable by its users, took the decision to disallow patent examiners to use the system due to the fear that it allowed the prior art and interpretation of past technology to be

[32] P. 237.
[33] See, e.g., EP1247168, 'Memory Shared Between Processing Threads', granted 2004.
[34] See, e.g., US 7,013,366, 'Parallel search technique for store operations', granted 2006.
[35] Knuth *Fundamental Algorithms*, p. x.
[36] This is a very common analogy used in teaching basic computer science.
[37] Proponents claim the advantage of having no syntax checking to do – the syntax is essentially the data structure. Any program algorithm is always partly embedded in the language/data structures of that language.

amended easily.[38] Critics of the USPTO noted that this was just the kind of user-based system which had been proposed by IBM and others to help solve the prior art searching problem.[39]

Stacks and queues are at the very lowest level of data format and to a very large extent the computing community has developed a relatively clear understanding of what the data representation is and what it means, at least when applied to relatively simple applications. Much of this common understanding has come from images which simplify the concept and give a 'core meaning' in discussions amongst programmers, and have also become the manner in which patent applications are prepared for computer-related inventions. These images have been used, too, in the description of algorithms where flowcharts show the step by step procedures involved in the program, albeit again usually at the abstract model level rather than detailing actual implementations.

Because the models are difficult to describe textually in any precise manner (which might explain the lengthier nature of software patent specifications), these images have become *ideograms* which both reflect the underlying object and also affect its meaning. Fleck used the notion of ideogram to describe the manner in which the conceptual thinking in anatomical drawing as medicine developed reflected the way in which the human body and its parts were drawn.[40] These were never 'photographic' likenesses but always reflected the 'thought style' of the medical artist and the importance which they gave to the different aspects of the body at that point in historical time. Computing diagrams are ideograms too; trying to represent the model with which the programmer is working but never producing a photographic reality of the underlying software system.

As an example of how the visual representation becomes a powerful computing metaphor, we can look to the idea of a 'software life cycle' which, depending upon the theoretical approach of the proponent using the ideogram, can be simple or more complex. A simple representation would be as in Fig. 2.6.

This represents diagrammatically a relatively ordered process of production, with each part of the process clearly separated from the others

[38] '"The problem with Wikipedia is that it's constantly changing", Patents Commissioner John Doll said. "We've taken Wikipedia off our list of accepted sources of information." An agency spokesperson said inquiries from *Business Week* about the use of Wikipedia led to the policy shift.' Reported, G. Aharonian, Internet Patent News Service, 26 August 2006.

[39] See, e.g., *The Peer to Patent Project: Community Peer Review of Patents* at http://www.cairns.typepad.com/peertopatent, which is supported by IBM.

[40] L. Fleck, *Genesis and Development of a Scientific Fact*, T. J. Trenn and R. K. Merton (eds.) (Chicago, IL, and London: University of Chicago Press, 1979).

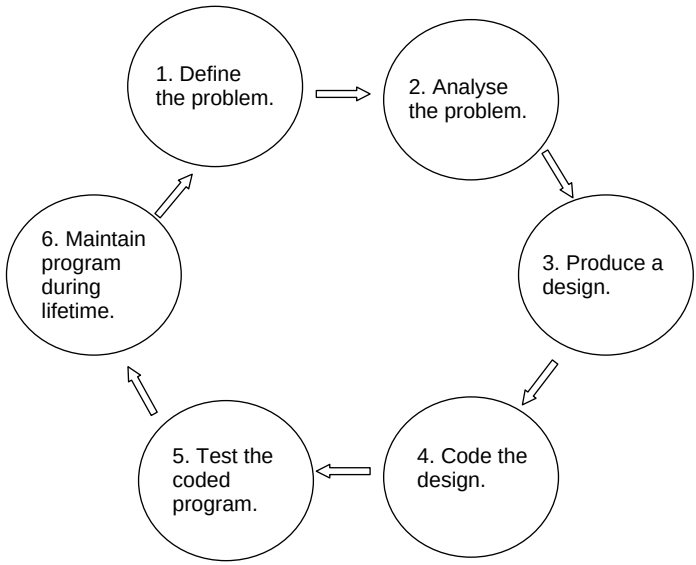

Fig. 2.6. The software life cycle

and generally of equivalent importance. This form of image was common in early textbooks on system production and certainly reflected the early view of the process, but quickly became the target of criticisms which suggested that the model being illustrated did not reflect the reality of the systems developer. Lehmann, for example, in a 1980 paper,[41] proposed that this simplistic image gave completely the wrong idea of the nature of development of a program, suggesting that it was machine-like:

> Laymen and professionals alike tend to perceive programs as 'mechanisms' that are conceived, designed and then simply constructed, that is, 'written', to solve some problem . . . It is generally accepted that the program as first visualised and eventually written will not be error free, that is, it will have to undergo a debugging process . . . but once bug-free it should be available forever to fulfil its purpose.[42]

At present, the diagrams used to represent the 'software life cycle' attempt (not always successfully) to integrate Lehmann's critical and now-accepted view of how software is 'evolved' rather than sequentially 'developed'.

[41] M. M. Lehman, 'Programs, Life Cycles, and Laws of Software Evolution', *Proceedings of the IEEE* 68(9) (1980) 1060–76.

[42] M. M. Lehman, and L. A. Belady, *Program Evolution: Processes of Software Change* (London: Academic Press, 1985).

In this ideogrammatic sense, a flowchart, as used in a patent application, is not describing the actual machine code, nor the program listing, nor how the program necessarily operates, but is describing an idealised view of how the patent attorney (based upon the description given by the programmer) views the operation of the program. An image of a data structure, too, is not really describing how the information is manipulated in the machine, but is describing an idealised view constructed by the attorney from the description provided by the programmer. In effect, these are abstract descriptions of the models which the programmer has tried to implement in code. The *Guidelines* for examination at the EPO encourage this strategy, by reminding the examiners that they are interested in the 'invention' rather than the specific implementation by program:

> In the particular case of inventions in the computer field, program listings in programming languages cannot be relied on as the sole disclosure of the invention. The description, as in other technical fields, should be written substantially in normal language, possibly accompanied by flow diagrams or other aids to understanding, so that the invention may be understood by those skilled in the art who are deemed not to be programming specialists. Short excerpts from programs written in commonly used programming languages can be accepted if they serve to illustrate an embodiment of the invention.[43]

Below, we discuss further whether it is possible to be 'skilled in the art' and yet not be a 'programming specialist'. What is important here is to understand why software descriptions are so different from those of, for example, chemical, engineering or electrical patents:

1. The underlying programmed code is usually ignored.
2. The programmer is working with virtual models which may be common (such as tables, stacks or trees discussed above) but which are in many areas likely to have been self-constructed by the programmer, or the programming team, or provided by other teams.
3. It is often the model from which the program is constructed which is at the heart of the 'inventive idea'.
4. Describing the model is usually done with diagrammatic techniques which are further abstractions of that model.

The effect of these points is that an invention can reside to a very large extent in diagrammatic representations of the model. This can cause interpretational problems because it is entirely possible for exactly the same underlying invention to be described but appear to be different entities. We can see this when we look at the language Prolog, which has been used in artificial intelligence programming. The model of

[43] Ibid., Chapter II, Content of a European Patent Application (Other Than Claims), section 4.14a, Computer programs.

	1	2	3	4	5	6	7	8	9
Make	**Cord**	**Cord**	**Cord**	**Reo**	**Reo**	**Reo**	**Duesenberg**	**Duesenberg**	**Duesenberg**
Condition	Running Well	Running poorly	Not running	Running Well	Running Poorly	Not running	Running Well	Running Poorly	Not running
Commission	5%	10%	10%	5%	10%	10%	Variable	Variable	Variable
Shop Work	Not required	One Week	Six Weeks	Not required	Two Weeks	Six Weeks	Estimate	Estimate	Estimate
Manager O.K.	Not required	Not required	Not required	Not required	Not required	Not required	Required	Required	Required

Fig. 2.7 A decision table structure[44]

[44] Redrawn from M. Montalbano, *Decision Tables*, Palo Alto, CA, 1974, p. 24, Science Research Associates.

processing used by Prolog is non-procedural,[45] rule-based and logic-based. A rule in Prolog might take the form:

```
if make of car is Cord
   and condition of car is running well
then
commission is 5%
and shop work is not needed
and manager is not required.
```

and be one of a number of rules which reflect both the data and procedure in the program – an example of non-procedural programming where the two are combined into one format (sometimes called 'declarative'). This is the basis of the rule-based expert system[46] which was particularly popular in the 1980s and 1990s[47] and was heavily promoted as an effective language for programming many task areas and as being a novel method of programming. However, it is really only novel when viewed as one particular representational model: the prior art by the 1970s certainly contained other programming forms which carried out exactly the same rule-based actions but did not use the logical *metaphor* that was used by Prolog. For example, program generators such as Cobol pre-processors did effectively exactly the same thing by using a table-format, rather than a rule-format as is shown in Fig. 2.7.

This is an example of two entire programming systems which have conceptual equivalence, but the problem also resides at the other end of the code continuum. For example, the GOTO and IF control structures have consistently been viewed as problematic in programming because they lead to badly structured programs, and badly structured programs are difficult to debug and to maintain. A solution to these was developed as the CASE statement – an improvement in programming technology, but one which is essentially identical to a well-structured GOTO or IF module of code. To a programmer, the CASE construct was a considerable technical advance: '[The case statement] was my first programming language invention, of which I am still most proud, since it appears to bear no trace of compensating disadvantage.'[48] Is it, though, an advance to those who see only that we are using two different expressions

[45] By which we mean that the definition 'program = algorithm + data' is not fully applicable. Examples of other such programs are query languages, interactive database programs and spreadsheets.

[46] See EP0782730, granted 2003 as an example of this rule-based expert system format.

[47] Critically discussed in P. Leith, *Formalism in AI and Computer Science* (London: Ellis Horwood/Simon and Schuster, 1990), ch. 3.

[48] C. A. R. Hoare, *Hints On Programming Language Design*, Stanford Artificial Intelligence Laboratory Memo AIM-224. STAN-CS-73–403 (1973).

which represent the same equivalent flow of control in a program, rather than anything more substantive?

Later – in Chapter 4 – the notion of 'metaphor' is looked at again with regard to the examination process, but for now it is important to realise that metaphors, as idealised structures or methods or abstract models, play a very large role in computer science thinking yet are relatively undefined in any theoretical manner. We might say that they provide an 'idealised construct' towards which the programmer attempts to push his coded implementation.

Looking back at the diagram in Chapter 1 from the Nymeyer patent, we can see just this sort of virtual model confusion in play. What is actually being protected? Certainly not the program code nor its particular implementation. What is being protected is an idealised model whose reality lies in the image constructed by the attorney – a mix and match of hardware, data structure and algorithm all put together to make a seemingly concrete entity. Only software would have allowed the attorney the total flexibility and freedom to produce such a series of diagrams to represent the 'invention'. The ideogrammatic aspects of this way of describing computing artefacts indeed goes further and allows descriptive worlds which are part technical and part ideological. This is particularly striking in the attempts by the artificial intelligence community to prove that their programs demonstrate 'intelligence'. Thus, one well-known early program (*AM*) was criticised for presenting a model of mathematical discovery which – according to critics – was only enabled to do such discoveries because these were incorporated within the program itself.[49]

It is generally accepted that, in patent matters, images have always been problematical, particularly in litigation. For example, Lord Reid in *C Van der Lely NV* v. *Bamfords Ltd*[50] pointed to the difficulties which lawyers traditionally had with photographs, noting that while they may be experts in the use of the English language, 'we are not experts in the reading or interpretation of photographs'. Lord Reid suggested:

[49] G. D. Ritchie and F. K. Hanna, 'AM: A Case Study in AI Methodology', *Artificial Intelligence*, 23(3) (1984). This is a particularly interesting case, since it is one of the few examples in the literature where a program's code was discussed in any detail by anyone apart from the author of the program. The program author, Douglas Lenat, disagreed with this assertion, believing that the program's model of mathematical discovery mirrored that of reality. AI 'invention' is an interesting notion; see the claim that the EPO has the ability to examine them in Y. Skularikas, 'Annex Regarding Experience Made When Examining in Particular Artificial Intelligence (AI) or Neural Network (NN) Cases' (Munich: EPO, 1994).

[50] [1963] RPC 61.

The question is what the eye of the man with appropriate engineering skill would see in the photograph, and that appears to me to be a matter of evidence. Where the evidence is contradictory the judge must decide. But the judge ought not in my opinion attempt to read or construe the photograph himself; he looks at the photograph in determining which of the explanations given by the witnesses appear to be the most worthy of acceptance.

If photographic representations can be seen to be confusing to a legal reader – given that they will show an actuality – then consider that the images which deploy notions of virtual worlds which have been created in the minds of programmers must be even more confusing. The programmer will know that some of these aspects outlined in the image will relate to structure, some to algorithm, some will reflect the way that the external world has been structured in the analysis phase, etc., yet it is not clear how a judge – even given the aid of expert witnesses – will be able to extricate all this information if difficulty is found with the reading of photographs, a visual technology which is part of the everyday world.

Textual descriptions

The language used to describe software inventions is important, since the application must be examinable as to novelty and inventive step, and if granted it must be accessible to readers. Thus the EPO guidelines require 'avoidance of unnecessary technical jargon, the use of recognised terms of art is acceptable, and will often be desirable'.[51]

Textual descriptions in computing, though, have a long history of falling short of requirements. For example, we have seen that the definition of programming languages has been problematical: anyone who has tried to program with a manual in front of them will realise that there is much which is never explicated by the manual or is written in a manner which leads to an incorrect assumption about what the programming construct does. Computer scientists have thus for many years attempted to find more formal methods of defining a programming language so that these confusions are removed.[52] One of the first attempts was with the Vienna Development Method (VDM), an example of the operational approach – that is, an approach which took as given that textual descriptions of programs were lacking in rigour and that the best way to describe a programming language was to provide a definition of how the memory

[51] European Patent Office, *Guidelines for Examination* (June 2005) Part C, Chapter II, 4.14, available online at http://www.european-patent-office.org/legal/guiex/e/index.htm.

[52] See, e.g., C. Jones, *Systematic Software Development using VDM* (Eaglewod Cliffs,' NJ: Prentice Hall, 1990), available online at http://www.csr.ncl.ac.uk/vdm/ssdvdm.pdf.zip.

of the computer was altered by each statement. An abstract definition of the language thus offered a shorter and more mathematically oriented description than text. These methods have their critics but the proponents continue (in the face of evidence, perhaps) to suggest that the techniques work. It is not simply those working in formal definitions who suggest that text offers poor descriptive facilities for something as complex as a computer artefact. All agree that the problem exists, but there is little agreement over there being an available solution. It is in this context that we must consider the written descriptions of software inventions in patent applications, remembering that it is the aim of the patent attorney to gain protection for their client, rather than to help move computer science towards a structured notational system.

The approach of the patent attorney which emphasises persuasion of the examiner seems to be generally found[53] and perhaps to be more possible in software-related inventions. Certainly, the many vociferous criticisms of software inventions relate to the obviousness which many computer scientists feel when they read a specification – particularly those which have been coming from the US Patent Office. In Chapter 3 we look in more detail at the policy elements of patenting computer-related inventions, but for now we should realise that the descriptive texts in the specification – the way that the invention is described and outlined – are very often at the heart of opposition to protection for this new technology. For example, the open-source proponent, Richard Stallman, has argued that the descriptive text is frequently pumped up to a level which the underlying idea does not justify:

> Programmers are well aware that many of the software patents cover laughably-obvious ideas. Yet the patent system's defenders often argue that these ideas are nontrivial, obvious only by hindsight. And it is surprisingly difficult to defeat them in debate. Why is that?
>
> One reason is that any idea can be made to look complex when analyzed to death. But another reason is that these trivial ideas often look quite complex as described in the patents themselves. The patent system's defenders can point to the complex description and say, 'How can anything this complex be obvious?'[54]

As an example of sheer complexity of description we can look at IBM's Commit patent – discussed in Chapter 1 – a relatively simple idea to the computer scientist who would be viewed as the patent addressee: a person

[53] See P. Leith, *Harmonisation of Intellectual Property in Europe: A Case Study in Patent Procedure*, vol. 3, Perspectives on Intellectual Property (London: Sweet & Maxwell, 1998) and Chapter 3 of this text.

[54] R. Stallman, 'The Anatomy of a Trivial Patent' (2000), available online at http://www.linuxtoday.com/news_story.php3?ltsn=2000-05-26-004-04-OP-LF.

skilled in the art of database system design. The patent specification is 109 pages in length and contains fifty diagrams. It is not necessary to follow Stallman and say that the patent is obvious, but to point out that what is essentially quite a simple idea has been certainly made to look very much more complex simply by the length of the description, and many readers will expend much time trying to find the inventive idea because they have to work through so much descriptive text. A textbook which dealt with an error-recovery operation of a similar nature and complexity to this Commit procedure would surely not use up such a considerable amount of space describing that operation: if so, textbooks would be so large as to be physically impossible to use. The Board of Appeal – considering only claims 20 and 21 – made no mention of the length of the patent application. Why might the board have had something to say? Because there are clearly policy reasons in having specifications which are concise: they can be more easily searched in future; they are more amenable to public use when the reader does not have to wade through page upon page of detailed implementation which involves trademarked hardware and software; and they better explicate the idea contained within the patent in litigation when clarity is to the fore. Article 83 of the EPC requires that the application 'must disclose the invention in a manner sufficiently clear and complete for it to be carried out by a person skilled in the art'. The critic of software patents would argue that when a specification is unduly long and obtuse, the clarity requirement of Art. 83 is not being complied with.

The skilled man

The patent system is based upon the understanding of 'a person skilled in the art' – the reasonably educated and experienced man who would be able to read a specification and understand it, and would have 'practical knowledge and experience of the kind of work in which the invention was intended to be used'.[55] The person skilled in the art is generally viewed as being without an inventive streak – he is a recipient of information and can use it in a practical manner, but cannot create it. Thus, he is a central pillar of the 'obviousness' criterion in examination –[56] which we look at later – but also if a specification cannot be used to recreate the invention from the person skilled in the art's reading, then revocation of the patent is possible.

[55] Lord Diplock in *Catnic* v. *Hill and Smith* [1982] RPC 183.

[56] Art. 56 EPC: 'An invention shall be considered as involving an inventive step if, having regard to the state of the art, it is not obvious to a person skilled in the art.'

In our nuclear handling experiment, who would be the 'person skilled in the art'? A nuclear site would have many engineers, scientists, accountants, managers who might all understand the use of the new programmed materials-handling system. If a patent was to be applied for, to whom would the document be directed and who would be the subject of the obviousness hurdle? This depends upon how we frame our 'invention' if we apply for protection. If we were to choose to present it as an invention which is entirely software-related, though, should our patent addressee be a programmer? If the addressee is a programmer, do we direct our description using programmatic techniques? The guidelines suggest that, '[i]n the particular case of inventions in the computer field, program listings in programming languages cannot be relied on as the sole disclosure of the invention', which indicates that we might use our program listing, but only as part of the application. An examiner who was able to program in one language might not be able to read the code in our language, but – on the other hand – it may be that the language which we are using is particularly adept at highlighting the inventive aspect (in fact, it may be an invention which is specifically related to an improvement in that language) which would require, rather than a short program listing giving a reasonably clear reading, a more complex mix of image and text to put across the same idea.

'Sufficient disclosure' in patent specifications is important but requires that the drafter of the document is aware of just who the addressee is. If sufficient information is not made available and the addressee could not implement the idea from the described invention, then revocation is possible. In *Halliburton*, Pumfrey J discussed this with reference to the potential failure of the patent being litigated to give sufficient disclosure, and noted that:

> if reference is made to the specification for a disclosure that enables the skilled man to construct a computer program to assist in the iterative technique claimed, or even enables him to carry it out manually, disappointment will be the result.

In many ways, *Halliburton* mirrors our nuclear example – a field which requires multiple areas of expertise brought together into one single program where the invention – if any – lies. Pumfrey noted that:

> The general use of computers in modern technologies raises particular problems, because the writing of anything other than a trivial program requires a substantial amount of effort in writing and debugging (programming's version of trial and error), even though much programming requires no creative thought and a competent programmer will be equipped with substantial experience in his area of expertise. When such a programmer forms part of the team which is the notional addressee of a computer-based invention, it is essential to form a view of his capabilities.

Pumfrey J's claim that much programming may not require 'creative thought' is perhaps to be viewed in the context of creativity at the level of inventiveness but, to this author's view, programming without creativity is simply not possible: one creates the world for each and every application through the use of data structure and algorithm. However, this is a clear statement that, where an invention resides in software, the level of skill of the programmer is important as one basis for sufficiency and obviousness, though the programmer may not be the only required individual when a team is involved, a situation which will be normal in many patentable fields. *Halliburton* does not really help us to allocate programming skill amongst the members of a team, since it focused upon the underlying mathematical model representing a drilling bit and its cutting action and – indeed – the patentee failed at first instance because of lack of detail in the specification.

Perhaps a more interesting example from the point of view of current software (particularly business method) applications is that from the Boards of Appeal. *Comvik* was an appeal against revocation of EP579655[57] after opposition. The patent had dealt with a reasonably simple idea: that a mobile telephone might have two identities associated with it rather than the usual one found in SIM cards. The Board of Appeal, considering Art. 56 in relation to whether the idea was obvious to the person skilled in the art, dealt with the nature of this person:

> 8. Finally, the identification of the skilled person may also need careful consideration. The skilled person will be an expert in a technical field. If the technical problem is concerned with a computer implementation of a business, actuarial or accountancy system, the skilled person will be someone skilled in data processing, and not merely a business man, actuary or accountant.

A main aim being taught by the patent was the allocation of costs with claim 1(iii) stating: 'the selective activation being used for distributing the costs for service and private calls or among different users.' The addressee of the patent should, the board decided, more correctly have been a GSM expert than someone involved with costings – since these allocations as such were not the formulation of a technical problem. A GSM technical expert would have been told of the requirement and would, the board suggested, have found the solution in the prior art which existed and was accessible by that technical expert. The board also pointed out that no new way of charging costs was disclosed in the document, and that only 'minor modifications of the network's home database' was required

[57] 'Method in Mobile Telephone Systems in which a Subscriber Identity Module (SIM) is allocated at least two identities which are selectively activated by the user', filed 1992.

which neither involved technical ingenuity nor a contribution to inventive step. We see in *Comvik*, therefore, that though the applicant's addressee may be a person with business-oriented skills, the moulding of applications to fit in with the 'technical' requirements of the EPO requires something more than just programming expertise. Yet compare this with the *Menashe* patent discussed in Chapter 1. It is not clear that there is any real technical implementation or novelty in the invention – well-tried and tested computing techniques are used to implement a gaming system on the internet. Where, we might wonder, is the person skilled in the art to be found here? A reader of *Menashe* might believe that the skill lies in the business of gambling rather than any specific technical skills: indeed *Comvik* and *Menashe*, though different fields, appear to have much in common as to their substance.

In the UK Patent Office *Comvik* has been discussed with regard to what might be the result if the person skilled in the art is a programmer.[58] The applicant in *NTT*[59] had suggested that:

> the person who would be concerned with solving the problems identified in this application would be a signal processing engineer, or an expert in data processing, and that therefore the problem that they have solved must necessarily be a technical problem.

Which the hearing officer found unreliable as a general principle:

> not least because if one develops the argument to its logical conclusion it would suggest that all computer programs must be patentable because they are written by technical people, i.e. people with data processing skills. Such a conclusion cannot be right, since the Courts have consistently said that computer programs that do not involve a technical contribution are not patentable in the UK. That would be a pointless thing to say if all computer programs necessarily involve a technical contribution.

We see the legal image of the nature of the programmer in these kinds of judgments and decisions. He is frequently viewed as a dolt who is given instructions and then implements them with little creative thought but perhaps much sweat of the brow – a mere 'data processing expert' with limited technical contributions to the problems they solve. Such an image of the programmer is certainly acceptable if one takes the machine-based perspective which, Chapter 1 argues, is to be found in the patent system's handing of software, but is not acceptable when one looks in more detail

[58] Note that the quaint title (from the 1960s) actually used in this decision was 'expert in data processing', a descriptor which certainly casts its own shadow over the programming profession.

[59] *NTT Communications Corporation Patents Ex Parte Decision* (O/195/05).

at what the programmer actually does and the nature of their expertise. Programming is certainly – at the professional level – as technical a task as that of designing a machine. The hearing officer's logic in *NTT* itself was flawed (in terms of common language) through mixing up 'technical' with the requirements of patentability. A program is certainly technical – all programmers are technical people – but that does not mean that all programs should necessarily be patentable – that is, that they make a 'technical contribution' which is sufficiently novel and inventive to be protected.

Can a software-related invention be understood outwith programming? The answer clearly depends upon the addressee of the patent: if it teaches a technique which is related to part of a programmed system (say, an improvement in operating system memory management), it would be inconceivable that it could be described without some level of virtual model which I have suggested is at the heart of programming. This must mean that in those circumstances the examination of the patent requires programming expertise. On the other hand, if the software invention is described at a different level of generality, then perhaps the programming model of examination is not required. We discuss this in the next section.

Where lies the invention? Levels of abstraction

Our nuclear materials-handling system – we think – is novel and inventive. The government are keen that development costs of the nuclear programme are kept low and suggest that we 'exploit our intellectual rights' wherever possible. We hire a patent attorney and suggest to him that he protects our 'invention', a suggestion which raises many possible tactics and methods. For example, an attorney might be interested in from whom we require protection – is it other commercial UK nuclear operators? Is it overseas, thus requiring foreign national patents? The tactics of protection are generally specific to each invention but the attorney is looking for as broad protection as possible, with the potential for maximising income.[60] Thus, rather than protect an inventive lock alone, the attorney would consider protection of lock and key as separate entities since more people will buy keys than buy locks.

[60] In fact, the commercial insight of the attorney has always been less effective than wished for. See P. Leith, *Harmonisation of Intellectual Property in Europe: A Case Study in Patent Procedure* (London: Sweet and Maxwell, 1998). There are various reasons for this: e.g. many ideas are not economically valuable and it is difficult to predict which will so be.

To our computing perspective, there are a variety of novel aspects about the system which we might propose as suitable for protection:

- the use of a computer to manipulate blocks;
- the program code for our interpreter;
- using a given data structure to represent nuclear materials;
- a clever algorithm to check speedily the validity of objects and placement;
- the design of our formal input language;
- an interpreter which carries out error-checking on block placement;
- the use of a program to control a mechanical arm;
- storing information about past interactions for later analysis;
- storing this historic information as a new artefact – a 'database' (as we decide to call it);
- the replacement of monkeys' mental operations by machine;
- the idea of a nuclear materials handling program.

Some of these may be protectable under the patent system and some may not and we look in the next chapter at the various reasons for protection or denial of protection. What is important here is to see that the kinds of the things we might call inventions are varied and wide ranging – perhaps more than in any other technical field. This is in large part due to the way that we can conceive programs: we view them as being different artefacts depending upon the level of abstraction we are discussing at a particular point in time. Thus, at the lowest level, we might want protection for the way that we have carried out a particularly slick implementation of our representational model. We might then look for protection at a higher level of abstraction, in the way that (for example) we have integrated the algorithm with the data structure. We can also at the highest level of abstraction, try to gain protection for the idea as a whole – the replacement, say, of monkeys with a system composed of a well-known machine and a novel program.

There are dangers in allowing protections for the most abstract of descriptions – since, at the most abstract what is potentially being protected is the problem itself rather than a solution to the problem. This criticism has become quite pronounced in the US, where the Federal Court of Appeals has – various commentators have suggested[61] – removed the need in software patents to have any substantive enabling

[61] See, e.g., D. L. Burk and M. A. Lemley, 'Policy Levers in Patent Law', *UC Berkeley Public Law Research Paper No. 135; Minnesota Public Law Research Paper No. 03–11* (2003). 'The Federal Circuit has essentially excused software inventions from compliance with the enablement and best mode requirements, but in a manner that raises serious questions about how stringently it will read the nonobviousness requirements.'

information at all. One IBM attorney[62] suggested that this approach has 'attracted the patenting of ideas which have been visualised as desirable but have no foundation in terms of the research or development that may be required to enable their implementation'. If this criticism is accepted, then it seems preferable that examination is carried out by those who understand the underlying programming methodologies and suchlike, rather than those who simply look at the abstract description provided.

A program is complex not just because it contains many lines of code which have been debugged, but because it contains so many different perspectives which represent the different virtual elements of the design, the implementation and the use of the program. This complexity gives rise to a whole host of possible inventive ideas, some of which are certainly novel and some of which are of questionable novelty arising more from the descriptive novelty (that is, the virtual model) perhaps than any real, new advance as perceived by other programmers.

Conclusion

In this chapter an attempt has been made to clarify the creative enterprise which is at the heart of programming: usually something which does not appear in the literature of the patent system. Programming has been very much set in the background and trivialised compared with other more mechanistic approaches, even though this has meant that the design of an inflatable kayak would have been viewed by the system as more worthy of protection than, say, a language which was so effective that it cut the number of semantic errors in program development substantially. Given the size of the software industry, such an invention would have huge economic benefits – more, we might imagine, than having cheaper inflatable kayaks.

Yet even if we take a programmer's perspective, our problems are not solved. The nature of software raises a whole number of issues about what it is that we might be protecting: we may want to protect novel software ideas, but how do we know that they are really novel and not just another way of saying or drawing the same idea that many other programmers have utilised before? James Moor[63] expressed these kinds of issues with regard to the claims to developments in artificial intelligence in the 1980s,

[62] J. D. Flynn, 'Comments on the International Effort to Harmonize the Substantive Requirements of Patent Laws', IBM submission at http://www.uspto.gov/web/offices/dcom/olia/harmonization/TAB42.pdf.

[63] J. H. Moor, 'Three Myths of Computer Science', *British Journal for the Philosophy of Science*, 29(3) (1978), 213–22.

albeit in a wider perspective. He pointed to three ways of describing a software artefact: theory, model and code. We have here been particularly referring to the model and code aspects.

To the lawyer, this notion of a similar concept differing at the various levels of abstraction may be difficult to comprehend – after all, the aim of good and clear legal thinking is to pin down concepts so far as possible. Programmers, however, do not take a similar view: the idea is not to fix any concept in concrete, but to view it as malleable, since this is where the power of programming arises.[64] Sometimes programmers look at the patent system and its desire to make concrete what should not – in their view – be so tangible, and conclude that if that is the only way in which the system can handle computing concepts, then it would be better that the system didn't handle it at all.[65] The developments within the EPO over recent years have begun to take more account of the programmer's view – but there is still a substantial way to go before software is as smoothly handled by the examination system as are the traditional fields of technology.

[64] We refer – in Chapter 5 – to Allan Newell's assertion that the 'models are broken' because of this malleability.

[65] This is the argument made by T. Tamai, 'Abstraction oriented property of software and its relation to patentability', in *Information and Software Technology*, 40 (1998), 253–7.

3 Policy arguments

Spectacular prizes much greater than would have been necessary to call forth the particular effort are thrown to a small minority of winners, thus propelling much more efficaciously than a more equal and a more 'just' distribution would, the activity of that large majority of businessmen who receive in return very modest compensation or nothing or less than nothing, and yet do their utmost because they have the big prizes before their eyes and overrate their chances of doing equally well.[1]

Introduction

The debate in Europe over the proposed Directive on Computer-Related Inventions has been heated.[2] To the Commission, the proposed directive was non-controversial, simply supporting the decisions of the Boards of Appeal of the EPO and integrating them within a European legal framework, since the EPO is an international body[3] rather than one controlled by the EC. Viewed in that light, it appeared to be a harmonising means to resolve 'uncertainty and divergences' in the protection of software across Europe, since by producing a directive, the *de facto* situation of patent protection for certain kinds of software patents would be harmonised across all member states, and such harmonisation has been the rationale of much of the EC legislative programme. The EPO position was the *de facto* position because the many important countries in terms of software production have had courts which have accepted the role and decisions of the Boards of Appeal and have, to a very large extent, agreed

[1] J. A. Schumpeter, quoted by F. M. Scherer, 'The Innovation Lottery', in R. Dreyfus, D. Zimmerman, and H. First (eds.), *Expanding the Boundaries of Intellectual Property* (Oxford: Oxford University Press, 2001), pp. 3–21.

[2] Proposal 2002/0047 of 20 February 2002.

[3] The EPO is formally controlled by its member states via the Administrative Council, the Council and the Office together comprising the 'European Patent Organisation'. Art. 4(3)EPC: 'The task of the Organisation shall be to grant European patents. This shall be carried out by the European Patent Office supervised by the Administrative Council.'

with the technical contribution approach outlined in Chapter 1: that is, if the invention looks 'machine-like', then it is worthy of protection.

The opposition to software patents[4] did not view the directive as a tidying-up operation, but rather saw it as the EC supporting an unlawful misreading of Art. 52 by the Boards of Appeal and indeed going further than the Boards of Appeal had already gone by encouraging broad protection for software *as such*.[5] Using highly effective campaigning techniques, the opponents succeeded in having the European Parliament review and suggest amendments to the proposed directive. A number of revisions were thus sent to the Council of Ministers, the joint approval of which was needed to enact the law. The Council of Ministers were not supportive of these revisions and a further directive was produced which once again reflected the earlier thinking. This led to further debate and argument, with the Parliament eventually voting against the directive (648 against 14). Software patent opponents suggested this was a victory – as indeed in some ways it was – but proponents of the EPO's approach could also claim victory over the attempts by the Parliament to limit the Board of Appeal's practice of protecting computer-related inventions when they have a sufficiently technical basis. The language used by the opponents of software patent protection was based on the notion that small software developers and the open source community were 'at war' with vested interests to protect their rights to innovate. For example, Mueller, in writing his history of his part in the opposition to the directive could claim:

> On July 6, 2005, the world of politics turned upside down. Big money was dealt a blow.
>
> The European Parliament threw out legislation that the world's largest IT companies badly wanted. Under the pretext of protecting inventors against plagiarists, it would have handed those giants sweeping powers over Europe's high-tech markets. An electronic roll-call thwarted the wicked plan in a matter of seconds, but that decision was preceded by years of intense fighting.[6]

[4] This is not to say that it is the only form of opposition – a variety of arguments have been put. See, e.g., J. Ernst, 'Software Patents Under the Magnifying Glass' (2005) at http://www.juergen-ernst.de/info_swpat_en.htm, who argues that the assumptions behind patenting are logically inadequate and contradictory when applied to software.

[5] See, e.g., M. A. Rossi, 'Software Patents: A Closer Look At The European Commission's Proposal', paper presented at 5th EPIP Conference European Policy for Intellectual Property, Copenhagen, Denmark, 10–11 March 2005: 'Indeed, while the Commission's Proposal is presented as a confirmation of the *status quo*, it is hard to deny that it will have deep consequences for the design of the regime of legal protection for computer-implemented inventions, given that its provisions can be interpreted as implying the extension of patentable subject matter toward a situation nearly indistinguishable from an express deletion of the "as such" exclusion of computer programs from patentability contained in art. 52(2) and (3) EPC. This is most unfortunate for at least a couple of reasons.'

[6] F. Meuller, *No Lobbyists as Such* (2006), p. 4, available at www.no-lobbyists-as-such.com.

'Wicked plan' or not, it can be argued that the situation with regard to protection has not changed: software patents are currently being protected post-6 July 2005 in exactly the same manner in which they were being protected pre-6 July 2005. A continuing and growing stream of applications is arriving at the EPO for protection of software inventions and thus the potential minefield of costly infringement litigation for smaller developers continues to grow rather than to recede.

If the arguments over the directive did not affect the specific procedure at the EPO, it may well have caused a wider reflection throughout the Commission: that is, on the notion which had been held as an article of faith up to that point – that the more intellectual property protection was introduced the better it was for the economic development of the EC. Such pro-protection perspectives had been found in the Bangemann Report,[7] which led to the introduction of the Information Society Programme in the 1990s. Bangemann had suggested that IP rights were essential to the development of the new information economy in Europe, and his thinking was followed in the introduction of, for example, data base rights in the Database Directive.[8] A more measured view on IP rights would have suggested that giving monopolies to some would effectively mean limitations on the economic development of others – a point which had been made with respect to industrial protection by Jacob.[9] As an example of the belief that 'more is better' in the field of IPRs, the view taken by the Commission to justify the introduction of the Database Directive had been that protection was required in order to compete with the US producers of information products. Yet when the industry was analysed[10] as to the effects of the directive, it was found that – if anything – there were fewer European products in the marketplace than prior to the introduction of the directive. It may be that we now have a Commission which is slightly more receptive to critical voices evidenced by the hearing on 'Future Patent Policy in Europe',[11] which received a particularly high response rate from SMEs and the IT industry. However, it is important to realise there has been no Damascene transformation: one reading of the initial findings still indicates a feeling that the patent system is good for industry and that the 'education' of SMEs, rather than

[7] *Recommendations to the European Council: Europe and the Global Information Society*, Brussels, 26 May 1994.
[8] Directive 96/9/EC of 11 March 1996 on the Legal Protection of Databases.
[9] R. Jacob, 'Industrial Property – Industry's Enemy', *Intellectual Property Quarterly* 1 (1997) 3.
[10] *First evaluation of Directive 96/9/EC on the legal protection of databases*, Brussels, 12 December 2005.
[11] Consultation and public hearing on future patent policy in Europe, DG Internal Market and Services, 'Hearing 12th July 2006. Preliminary findings: issues for debate'.

a re-engineering of the protection system, will help resolve their worries and fears.

Another effect of the debate over the software directive is certainly that the desire of the EC to gain control over the European Patent Office has been increased. It is commonly held – by opponents of software protection – that the EPO is a creature of the Commission, but this is not true, there being more than a modicum of friction between the two.[12] The EC has very limited input to the operation and control of the EPO, merely being an observer at the Administrative Council along with WIPO[13] and other organisations. The EPO and the EPC were 'temporary' creations intended to be subsumed within the EC's Community Patent project but the continuing failure to resolve the issues of translation costs and litigation forum has allowed the EPO to become a large and powerful organisation with its own view of where it stands as an independent agent[14] in the world ordering of patent organisations – a view which may not fit the Commission's view of where the EPO should stand.[15]

This text is not an analysis of the politics of legislating for patent protection and assumes that, anyway, the procedural aspects of patenting are every bit as important as the legislative. These procedural aspects include that the Boards of Appeal have the necessary mechanism to impose their interpretation of the EPC and their technical assessment of what is protectable technology, through, for example, control of the large flow of patents into the national phases. But given that parallel to the development of protection for software there has been opposition to patent protection and alternative models offered, it is important to investigate the policy framework which underlies protection since that has certainly informed the procedural developments.

[12] There is also friction between the EPO and its Administrative Council. In interviews with EPO staff, the author was consistently told that no other patent office would have its competitors on its board of management. EPO staff saw the other patent offices as second-rate competitors, not collaborators. P. Leith, *Harmonisation of Intellectual Property in Europe: A Case Study in Patent Procedure* (London: Sweet and Maxwell, 1998).

[13] In fact, WIPO is specifically mentioned in Art. 30 EPC as an observer, whilst the EC is not.

[14] Interviews carried out with some Appeal Board members at the OHIM – the European Trademark Office – which is an EC institution rather than an international one demonstrated a high level of envy of Board of Appeal independence at the EPO. See P. Leith, 'Judicial and Administrative Roles: The Patent Appellate System in a European Context', *Intellectual Property Quarterly* 1 (2001) 50–99.

[15] FFII – who take an anti software patent stance – report that leaked EC documents suggest: 'the Commission criticises the European Patent Office (EPO) as a "business culture of its own" that considers EU interferences in patent law "as an attack on the holy writ".' The Commission also raises concerns that the EPO is 'assuming a policy role which does not belong to it'. See http://wiki.ffii.org/. (Dated 24 April 2006.)

Other protections

That patent protection is required for software is posited on the supposition that there is something worth protecting which is not currently being protected by the other forms of intellectual property law – particularly copyright, which has recently been, via TRIPS,[16] applied as the primary worldwide mechanism of software protection.[17] Looking at our nuclear materials-handling program from Chapter 2, we should be able to see which aspects a software innovator might want to protect but which are not given protection under the copyright regime. Despite any gaps it may offer, the copyright system is highly rewarding – lasting for seventy years after the death of the longest-surviving author. In a programming team, the youngest twenty-year-old author may live another seventy years, which could give protection of the source and object code of 140 years to the employer.[18] Yet, given this significant long term of protection, there are certainly weaknesses insofar as we might want to protect our inventiveness via copyright.

The directive on the protection of computer programs which provided a harmonised European framework of copyright protection, for example, specifically excludes the design of our programming language from protection:

> Protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.[19]

Where the preamble has included programming languages (and, importantly, algorithms) as idea and/or principle:

> Whereas, in accordance with this principle of copyright, to the extent that logic, algorithms and programming languages comprise ideas and principles, those ideas and principles are not protected under this Directive.

[16] Which protects software as a literary work: 'TRIPS Art. 10(1): Computer programs, whether in source or object code, shall be protected as literary works under the Berne Convention (1971).' There is debate over whether TRIPS should impose a requirement for patent protection for software, but countries have not amended patent laws to include this. See, for an overview of approaches at the time of the TRIPS agreement, Strauss J, 'Bedeutung des TRIPS für das Patentrecht', *GRUR Int* (1996), 179–205.

[17] Trade secret protection is also available and usually effected through control over previous employees. Many of the early software cases appear to have involved some accusation of theft of know-how.

[18] Authorship and first ownership are usually combined rights except when the author is an employee. For example, see s. 11 of the UK Copyright, Designs and Patents Act 1988.

[19] Article 1(2), Directive of 14 May 1991 on the Legal Protection of Computer Programs (91/250/EEC).

Though much effort and creativity may have been applied in the design of such a simple but powerful materials-handling programming language, if we follow the directive the copyright system is not appropriate for protection. The functionality of programming languages does appear to be the underlying issue here: despite the directive suggesting that a programming language is an idea or principle, and since ideas are not protected under copyright then neither should programming languages be so protected, it has been suggested that the real reason for exclusion of protection is more to do with a belief that languages are an essential tool which should not be monopolised.[20] This lacks clarity of reasoning: there are many hundreds, indeed thousands, of programming languages and if we take programming systems (such as the so-called 'expert systems', database packages, etc.) which allow programming, then we have in reality a situation where users have to be persuaded to use these, rather than a situation where users are fighting to be allowed access to a monopolised language. As Howard Aiken was quoted as suggesting,[21] in the computer field the problem 'was not to keep people from stealing your ideas, but to make them steal them'. The slow acceptance of, for example, object-oriented technology (developed through languages such as SmallTalk) is an example of just how much persuasion is required before the programming community will move to what is now perceived as an essential and very powerful programming technology. It may be that what is problematical is when a proprietary language becomes a standard; but there are many proprietary standards in computing and these do not appear to be viewed as negatively as programming language design.[22]

Lai has discussed[23] copyright protection in the UK context with respect to the doctrine of *merger* in the US, noting that when the underlying idea of

[20] D. Bainbridge, *Introduction to Computer Law*, 5th edn (Harlow: Pearson Education, 2004), p. 50, says: '[t]he exercise of rights in languages could seriously interfere with the licensing and distribution of computer programs and databases. In principle, there is a strong argument for saying that programming languages are ideas and, as such, cannot be protected by copyright.'

[21] Reported (p. 240) in I. B. Cohen, *Howard Aiken: Portrait of a Computer Pioneer* (Cambridge, MA: MIT Press, 1999). Similarly, Microsoft's position regarding piracy: 'If they are going to get pirated, you want them to pirate your stuff, not your competitor's stuff. In developing countries, it is important to have a high share of the piracy software.' B. Schneier, *Secrets and Lies: Digital Security in a Networked World* (Indianapolis, IN: John Wiley, 2004), p. 253 (brought to my attention by Yu-Lin Chang).

[22] Cohen has suggested that the same justifications which are used to deny protection to algorithms, scientific principles and languages 'apply to software platforms'. This demonstrates the pragmatic difficulty in dividing elements of software into those which justify protection and those which do not. See S. A. Cohen, 'To Innovate or Not to Innovate, that is the Question' *Mich. Telecomm. Tech. L. Rev.* 5 (1999), 1.

[23] S. Lai, *The Copyright Protection of Computer Software in the United Kingdom* (Oxford: Hart Publishing, 2000). See Chapter 3 in particular.

a given work has only a given number of possible ways of being expressed, then there is said to be a merging of the idea and the expression. This doctrine is applicable to standards as well, since if one wishes to conform to a standard (such as a menu format) then one is, the reasoning goes, following an idea rather than simply an expression. The application of this kind of reasoning is not particularly easy, since clearly it conflicts considerably with non-functional artefacts such as plays, TV screen formats and suchlike, where one could argue that more protection is provided when the idea and the expression are a close fit, requiring more effort on the part of those wishing to emulate the successful artefact to put conceptual space between their competing play or screen format. In the UK, Jacob J in *Ibcos*[24] rejected the merger doctrine of the US and suggested 'the true position is that where an idea is sufficiently general, then even if an original work embodies it, the mere taking of that idea will not infringe. But if the "idea" is detailed, then there may be infringement. It is a question of degree.' Clearly, the 'idea' behind a given programming language is important – perhaps utilising a novel processing model such as Lisp's use of functional syntax and semantics, or Forth's use of stacks – and may not be protectable, but the actual implementation of the language must be more than an idea. Jacob's reasoning in *Ibcos* would appear to suggest that our language is indeed protectable, since the expression of the idea is highly detailed – and this follows the reasoning underlying the copyright system.

Bainbridge has pointed to an unreported decision, *Microsense*,[25] relating to a programming system based on mnemonics, where the alleged infringer utilised forty-nine of these in a similar control system to that of the developer, arguing that there could be no copyright in them because of their functional necessity. Unfortunately for commentators, no decision was issued, but the judge indicated that there was an arguable case for protection. Our program has highly limited input facilities; unsurprising, given that it utilises a teletype keyboard. However, we might suggest that the commands we have chosen are copyrightable: such input as 'PUT BLOCK6 ON BLOCK2' is certainly – to someone with typing skills – as user-friendly as 'drag and drop' interfaces based upon pointing devices. Might this be protectable under the reasoning in *Ibcos*? If the later *Navitaire* v. *Easyjet* judgment is good law, then it seems unlikely, since Pumfrey J conflated the two ideas of program commands and program

[24] *Ibcos Computers Ltd* v. *Barclays Mercantile Highland Finance Ltd* [1994] FSR 275.
[25] *Microsense Systems Ltd* v. *Control Systems Technology Ltd*, 17 June 1991. See Bainbridge, *Introduction to Computer Law*, p. 51.

language together under the exclusion in Art. 1(2) of the Software Directive:

> In my view, the principle [of Art. 1(2)] extends to *ad hoc* languages of the kind with which I am now concerned, that is, a defined user command interface. It does not matter how the 'language' of the interface is defined. It may be defined formally or it may be defined only by the code that recognises it. Either way, copyright does not subsist in it.

While superficially this suggestion is reasonable – especially given the text of the directive – it does lead us into other potential problems when we try to set a boundary between commands and program. For example, it could lead to a situation where we protect a program when it is compiled (that is, translated into object code and then executed) but do not protect it when it is interpreted (that is, each line is translated as a single command). Pumfrey's interpretation of Art. 1(2) seems reasonable only because he is including the user in the picture: when a user inputs commands, he seems to be suggesting, it is a different situation from when a user has pre-prepared the commands in the form of a program. This is an echo of our discussion in Chapter 1, when the autonomous 'machine' view was taken of software, a view which suggests that software when part of a machine (e.g. compiled) is different from when it is not loaded onto the system and thus not part of the machine.

If it looks unlikely that we can rely upon copyright protection for our language design and commands, what about the compiler/interpreter which translates our user input into a form which checks for errors and, if valid, issues controls to the mechanical arms? Here things appear slightly more positive: our program is protected from literal copying of both the source code and the object code. For the idea behind the program, we have less protection: thus, though we are protected from *copying*[26] of the actual code, and also protected from much reverse engineering,[27] we have no protection should a competitor like our idea and implement a similar system based on our program's black box operation. This was the issue in *Navitair* v. *Easyjet*, in which Pumfrey J noted that software differed from other copyrightable artefacts by suggesting that 'two completely different programs can produce an identical result:

[26] Note that the source code and object code must be in some way accessed – there cannot be copying if we have not had access to the original.

[27] One of the many controversial aspects of the Software Directive was Art. 6, which allows decompilation (that is, reverse engineering) only for the purposes of interoperability. For a review of reverse engineering and software from the US and anti-reverse engineering perspective, see P. Samuelson and S. Scotchmer, 'The Law & Economics Of Reverse Engineering', *111 Yale L. J. 2002, 1575–663*.

not a result identical at some level of abstraction but identical at any level of abstraction. This is so even if the author of one has no access at all to the other but only to its results.' The case involved two companies: Navitaire had produced an airline reservation system and EasyJet had been a licensee of this program. EasyJet wished for its own system – though based on identical functionality to that of Navitaire – and used clean room techniques (that is, purely observation of the operation of the other system rather than its code) to produce an 'identical' rival system. Pumfrey J found for EasyJet, indicating that our own novel program could similarly be copied by our competitors using such techniques. We can say, therefore, that the copyright system will certainly protect our effort from plagiarism, but will not protect the implementation of a similar idea. The reverse engineering prohibition can be seen, perhaps, as a means to protect from direct copying, but does little to protect re-engineering of our product unless we have some functionality hidden in the code which is not visible through black box techniques.[28]

It is important to remember that we are discussing copyright protection: the author is normally central to processes of producing a copyrightable work which is fixated,[29] yet here we seem to be in a situation where copyright is only being awarded when the author has pre-recorded his work in a certain format (that is, as code which will be executed at a later date rather than input as command statements). It is as though languages and interpreted commands are being viewed as performances, rather than works of authorship. Of course, the analogy quickly breaks down since performances which are recorded are protected, whilst our languages and commands appear to receive no protection at all.

This is not a text on copyright of programs, but the point has to be made that there are weaknesses in the theoretical models of software copyright which are not being addressed by European courts or legislatures. There is something of a rag-bag approach to the issue of expression, idea and function – creating hybrid forms of protection on a case-by-case basis with little analysis of the underlying software issues. Some have, indeed, argued that there is a developing trend towards a conceptual hybridisation of copyright and patent protection anyway, which has been brought on by the development of software. (Such hybrid approaches have also been proposed as positive advances by some advocates of

[28] The technical term frequently used for investigation which allows internal observation of code is 'white box' or 'glass box' technique. It is most usually used in software testing and debugging.

[29] One of the requirements for copyright is that there is a recording made. This is certainly the case when commands are input to a machine – what happens in computer hardware is essentially a form of continual copying from one location to another.

alternative protection and we look at these in Chapter 6.) Burk has suggested that a 'stretching' has occurred as the legal systems try to accommodate software as both expression and function. Burk's argument transcends our immediate interest in that he has, for example, suggested that it is possible this hybrid conceptualisation will seep out into other areas of law; for example, in the US interpretation of the First Amendment with respect to program code.[30] Of course, that legal concepts transcend the field in which they develop and affect new areas in unexpected ways is not new, being one of the main thrusts of the argument by the realists in their criticism of formalist approaches to law.

Our program is further novel – we would argue – in that it produces a database of materials-handling operations: the SAVEMOVES command being the operation which writes the moves taken to off-line storage. Might protection for the output from this facility be available via the Database Directive?[31] Clearly, the *sui generis* right developed by the directive does not protect the idea of utilising a database, its aim being to protect those elements of the contents of a database which are otherwise not-protectable (e.g. a compilation of facts). But such protection may be useful to us because the long-term data could be analysed in a number of ways to help develop a better materials-handling program. Under the reasoning in *Mars* v. *Teknowledge*,[32] which gave protection to data descriptions of coins within a program as a 'database', such data would seem to be protected. However, the ECJ has more recently limited the right in three referrals which it dealt with concurrently,[33] proposing that the development of databases which arise directly from other processing does not give rise to a protectable database under the directive. Thus, our database of material moves is probably not protectable and our competitors – should they gain access – could utilise such data in the development of their own programs.

[30] D. L. Burk, 'Patenting speech', *Texas Law Review 99 (2000), 1–55*. In brief, freedom of expression is protected under the 1st Amendment, and if software is 'speech' then it too would benefit from a weakened ability of the US Government to control it – as, for example, in attempts to prevent dissemination of cryptographic software.

[31] Directive 96/9/Ec of the European Parliament and of the Council of 11 March 1996 on the Legal Protection of Databases. Article 7(1) deals with the *sui generis* right: 'Member States shall provide for a right for the maker of a database which shows that there has been qualitatively and/or quantitatively a substantial investment in either the obtaining, verification or presentation of the contents to prevent extraction and/or re-utilization of the whole or of a substantial part, evaluated qualitatively and/or quantitatively, of the contents of that database.'

[32] *Mars UK Ltd* v. *Teknowledge Ltd* (1999) 46 IPR 248.

[33] C-338/02, C-46/02 and C-444/02, 9 November 2004.

The argument for patent protection

At its most elemental, one primary argument for patent protection is made by simply looking at the gaps in the protection which is offered by the copyright system to our novel and inventive nuclear block-handling program. Clearly, the effort in coding is rewarded: that is, the design and implementation of our functional artefact, but not the *inventive input* upon which they are based. Given that software is one of the many important industrial products, why should invention in that industry be refused when other industrial products are protected?

Software is certainly a major industry, but one upon which it is particularly hard to put an accurate total European figure. This is due to its nature, as packaged goods bought off the shelf, one-off products produced by external software houses, and internally produced and maintained software. Indeed, even defining what is 'software' is a problem for analysers of economic data in their attempts to collect transnational data – a problem of definition caused when, for example, a software product is turned into a hardwired chip and becomes part of a more traditional physical product (for example, part of a car engine control system or DVD player).[34] However, it is generally seen as an important industry, and one where Europe is viewed as a large consumer – purchasing around 30 per cent of the world's software output.

The wider background to the political view of this trade – as already mentioned – was the Bangemann Report, which was viewed as a wake-up call to European industry. In the 1970s and 1980s much traditional industry had left Europe for the Far East and the worry was that there was no new industrial enterprise filling the gap. These worries were increased when Japan announced it was funding a Fifth Generation Computer Systems project, a project which was intended to increase the innovation in Japanese industry and move it beyond the cheaper copying of European-designed goods.[35] The response of European countries was to announce that they would fund projects to counter this

[34] See National Research Council, *Measuring and Sustaining the New Economy, Report of a Workshop* (Washington: National Academy Press, 2002). Also of interest is the attempt to use patent documentation to discover the extent of embedded software: see D. H. McQueen and H. Olsson, 'Growth of Embedded Software Related Patents', *Technovation* 23 (2003), 533–44.The picture presented cannot be entirely accurate and relates to Sweden, but the authors demonstrate quite substantial growth across several manufacturing and communication industries for products where software and processor are linked together as a hardware product.

[35] The current threat is China: 'Furthermore, global suppliers in the IT market have been required to reveal the source code of their software and submit their intellectual property

'threat'.[36] In the UK, the funding was provided under Alvey[37] and gave support to a variety of novel approaches including artificial intelligence, formal methods and parallelism. None of these funded programmes was particularly successful, and they did nothing to reduce the spectre of a Europe bereft of industry, but did perhaps give rise to a realisation that central control and development of technology was more difficult than governments had imagined and that it might be better to leave this to industry itself.

Bangemann thus proposed – with his committee of experts – a 'call to arms', that Europe may have lost its heavy and dirty industries, but not all was lost: the new 'information industries' would be the new economic provider in Europe. This information industry would be based on content and knowledge – Europe may have lost its steel foundries and its shipbuilding, but it still had cultural industries and an educational system producing highly educated and information-literate graduates. Importantly, the lesson the report said had to be learned from past experience was that centralised state control with centralised funding would not work, rather that the state should provide the legal framework in which the new industries could grow through individual business success. As Bangemann argued:

- it means fostering an entrepreneurial mentality to enable the emergence of new dynamic sectors of the economy;
- it means developing a common regulatory approach to bring forth a competitive, Europe-wide, market for information services;
- it does NOT mean more public money, financial assistance, subsidies, dirigisme or protectionism.[38]

to Chinese experts. Despite concerns by foreign firms, it is unlikely that the government will abandon its objective to create a strong national electronics industry with broad access to leading technology.' *European Competitiveness Report 2004*, p. 258.

[36] E. Feigenbaum and P. McCorduck, *Creative Computing* 10(8) (1984), 103, available online at http://www.atarimagazines.com/creative/v10n8/103_The_fifth_generation_Jap.php: 'The Fifth generation: Japan's Computer Challenge to the World', 'The Japanese are planning the miracle product. It will come not from their mines, their wells, their fields, or even their seas. It comes instead from their brains. The miracle product is knowledge, and the Japanese are planning to package and sell it the way other nations package and sell energy, food, or manufactured goods. They're going to give the world the next generation – Fifth Generation – of computers, and those machines are going to be intelligent.'

[37] F. D. Clery, 'Alvey: The Betrayal of a Research Programme', *New Scientist*, 1768 (11 May 1991): 'This week saw the publication of the final report of the outcome of Britain's Alvey programme of collaborative research in information technology in the mid-1980s. The report tells a familiar tale of first-class research languishing from lack of investment by industry to bring products to market . . .' My own view is that the academic community, rather than industry was to blame for the project failures. See P. Leith, *Formalism in AI and Computer Science*, (London: Ellis Horwood/Simon and Schuster, 1990), pp. 40, 91, 166.

[38] *Recommendations to the European Council: Europe and the Global Information Society*, Brussels, 26 May 1994.

What could the state provide as a framework if not subsidies or protectionism? For one thing, it could provide intellectual property rights, since: 'Creativity and innovation are two of the Union's many important assets. Their protection must continue to be a high priority, on the basis of balanced solutions which do not impede the operation of market forces.' For some ten years until the late 1990s, Bangemann was one of the most influential figures in technology and industrial thinking of the European Commission, a time when intellectual property rights were being favourably pushed by both the Commission and Parliament. Intellectual property rights clearly fitted the Bangemann bill – they were a means of encouraging private enterprise to innovate but kept the state at arm's length. This is not a novel approach: Kaufer has described how the Venetian state, in 1474, under financial pressure due to war and other costs but still keen to support innovation (at low cost) produced a patent code which made provision 'for the works and devices discovered by such person, so that others who may see them could not build them and take the inventor's honor away', so that 'more men would apply their genius, would discover, and would build devices of great utility to our commonwealth'.[39] Patent rights fit this approach well: the costs of development (and the cost of patent administration) are placed upon the innovator but the state benefits from that innovation. The state even has the opportunity to receive income above expenditure from patent offices.[40] Significantly – as we outline below – for the software patent debate, one aspect of this new policy for the information society was that it tended to encourage larger enterprises to be constructed: cross-national providers of telecommunications etc. were viewed as being more vital than smaller, national providers. In part the success of the US was perceived as arising from the size of its firms and this was something that the new approach wanted to encourage in Europe.[41]

This Bangemann approach of focusing on supportive frameworks rather than subsidies continues to date. For example, in late 2005 another

[39] E. Kaufer, *The Economics of the Patent System* (London: Harwood Academic Publishers, 1989), p. 5.

[40] All national offices appear to suffer this financial loss of income to the state.

[41] For example, national media ownership rules were to be reviewed: 'In addition to ownership controls to prevent monopoly abuse, many countries have rules on media and cross media ownership to preserve pluralism and freedom of expression. In practice, these rules are a patchwork of inconsistency which tend to distort and fragment the market. They impede companies from taking advantage of the opportunities offered by the internal market, especially in multimedia, and could put them in jeopardy vis-à-vis non-European competitors.'

programme[42] was being set up where intellectual property rights again are seen to play an essential role in the 'I2010' initiative to make Europe 'the world's leading knowledge economy in collaboration with developments of industrial policy'.[43] Such an approach is *IPR-active* in contrast to the pre-1990s relatively passive approach to intellectual property rights, since it seeks to develop IPRs in ways which will encourage and develop new industries – including new rights – whereas previous governmental approaches were more concerned, perhaps, about ensuring that the existing rights were effective and well managed. Is such an approach justified? Perhaps, but it might be that once again we can see that governmental intrusion to encourage innovation is not as easy as its proponents would wish. Despite a growing supportive framework, Europe still continues to drag behind the US in economic development, with a widening gap in GDP since 1990 evidencing a potential return to pre-1970 levels (the after-effect of the Second World War) where this had been 37 per cent in 1960 and in 2005 had fallen from 25 per cent (early 1980s) to around 30 per cent.

Bangemann's project is at core a liberal, free-trade recipe for supporting and building capital and enabling the development of *large* transnational enterprises, free from independent member state control and this core perhaps reads – to some eyes – as the very converse of democratic socialism. Hence, to remove this hard market-led edge, the report promotes the view that social advance and social development will be enabled through the market-led revolution. The regulation required is not that for cultural or social purposes – only to remove factors which might 'impede the operation of market forces'[44] and that regulation of property rights have to be seen in a global context in order to achieve success of the new information economies[45] which are based upon novel and inventive applications of the new information technologies.

[42] Communication from the Commission to the Council, the European Parliament, the European Economic and Social Committee and the Committee of the Regions, 'I2010 – A European Information Society for growth and employment', SEC(2005) 717, COM(2005) 229 final.

[43] Enterprise and Industry Press Release, 5 October 2005.

[44] It is interesting to see that the author's students today find bizarre the very idea that there could be only one supplier of telephone systems, or that a commercial service such as a mobile phone would be tied to the state. This is perhaps one indication of the success of Bangemann's deregulation and market revolution, based of course on the Thatcher reforms of the UK.

[45] 'The global nature of the services that will be provided through the information networks means that the Union will have to be party to international action to protect intellectual property. Otherwise, serious difficulties will arise if regulatory systems in different areas of the world are operating on incompatible principles which permit circumvention or create jurisdictional uncertainties.'

How does this fit in with the patent system? Clearly, it requires that the EU becomes party to supporting the internationalisation of patent rights. It implies that patent rights become an important element of the framework for these new economies. And it implies that patent rights in the new information economies should be available to and taken up by the information industries. This, then, was – and largely remains – the European context and which goes far to explaining why the proposed Directive on Computer-Implemented Inventions appeared to be a rational and reasonable response to the needs of industry. The rhetoric of the patent system has always been that it rewards innovation and forward-thinking businesses and, to any governmental organisation concerned about future economic prosperity, it must appear anathema to ignore the patent system as a means of economic development: supporting invention and wealth creation in Europe.

Such a programme also requires the active support of the EPO, a support which the EPO has been happy to provide. For example, in the 2004 Annual Report, the President noted:

> By stating in their Lisbon Declaration that they aimed to make Europe the most competitive knowledge-based economy in the world by 2010, the European Union's heads of state and government turned innovation protection and technology transfer into key items on the political agenda of the coming years . . . As an essential engine of knowledge transfer and a champion of effective innovation protection, the European Patent Office is aware that it has a major part to play in implementing the Lisbon Strategy. There is no other institution in which the strands of invention, innovation and commercial exploitation merge to the same extent, and there is no clearer proof than the doubling of annual filings between 1996 and 2004.

A question worth considering is whether this political rhetoric of support has affected or might affect the way that intellectual property is conceived and interpreted by the courts and other organisations such as patent offices? A legal formalist would deny the possibility that political context could substantially affect the legal interpretation of documents, but for many non-formalists[46] it does indeed appear possible that courts will pick up on the dominant pro-IP policies of the politicians as much as the literal meaning of the legislation. We can look to the US for one reported example of this, where Landes and Posner suggest that the US Court of Appeals for the Federal Circuit (CAFC)[47] and its highly pro-patent

[46] I have yet to meet a legal practitioner, who viewed their own cases through a formalist lens, though have met those who seek an ideal formalist solution – sometimes through logic.

[47] Created to harmonise patent appeals in 1982.

approach has avoided public controversy because it has followed the 'general drift toward expanded protection of intellectual property' and that the fact that there has been no public controversy is evidence of leaning in favour of patent validity in hearings[48] despite substantial academic and legal criticism.

Many proposals for harmonisation of the European patent system involve some kind of central court of appeal which would follow the role of the CAFC, perhaps based upon the European Court of Justice, a court which has been seen to be happy to extensively develop the rights available via trademark protection in line with the perceived needs of rights owners.[49] Would a European Patent Court or some such develop patentee's rights in a similarly expansive manner?

An important element, too, in the encouragement of patenting is that it implies that European companies will – in order to use the patent system – expend more effort upon research in order to gain patents. A clear and direct link has been evidenced by all economic studies to show that R&D activity and patent activity are connected.[50] Unfortunately, the evidence to date is that Europe's non-public R&D expenditure continues to lie well below that of its main developed competitors with Europe expending only 1.3 per cent on R&D in 2000 whilst Japan spent 2.2 per cent and the US 2 per cent.[51] By encouraging involvement with the patent system – from all sizes and types of industry – it can be expected that R&D expenditure would be required to rise, which would be to the positive benefit of industry and European economy.

Finally, the changing nature of industry within the EU, where manufacturing output and employment continue to fall, but employment and output in the software industry rises, gives a situation where what is being protected is, perhaps, no longer the prime product in terms of the European economy.

[48] See W. M. Landes and R. A. Posner, *The Economic Structure of Intellectual Property Law* (Cambridge, MA: Belknap Press, Harvard University Press, 2003), p. 336. Their findings are that the academic criticism is justified.

[49] For example, in exhaustion matters – Case C-355/96, *Silhouette International Schmied GmbH & Co. KG* v. *Hartlauer Handelsgesellschaft mbH*, 16 July 1998. See the discussion in *Select Committee on Trade and Industry, Eighth Report 1999 Trade Marks, Fakes And Consumers*, online at http://www.parliament.the-stationery-office.co.uk/pa/cm199899/cmselect/cmtrdind/380/38007.htm.

[50] A good review from a consistent research interest is that of F. M. Scherer, *Patents: Economics, Policy and Measurement* (Cheltenham: Edward Elgar, 2005).

[51] See *European Competitiveness Report 2004*, Commission Staff Working Document, SEC(2004)1397.

The policy argument against patent protection

There are several primary policy arguments against patent protection for software (excluding the 'unlawful' argument regarding Art. 52). These include:

- Patents hinder the software industry rather than invigorating it.
- Patent offices are either incapable of, or have great difficulty in, examining software patents, which leads to poor quality patents which will hinder the European industry (both commercial and open source).
- Patents are not suitable for SMEs and other smaller producers, who comprise a large part of the software production market, and thus will hinder European industry (both commercial and open source).
- Software is special because it involves 'network effects' which can be undermined by monopoly rights.

A further argument, not usually developed by the anti-patent lobby, but outlined below is that much of computing inhabits areas which are ideologically rather than technically motivated: in particular, those who argue for 'intelligence' being a technical characteristic of computing artefacts.

The first argument – the hindering one – is disputed, but many who support software patenting agree with the latter three of these arguments: some of these proponents would suggest that means can be found to improve patent office examination and limit rights to make them more palatable to SMEs. Indeed, on the third point – relating to SME suitability – for many of the IP litigation community in the UK, these arguments are held to be basically true and simply a fact of life which they feel is unlikely to be changed. In research interviews with IP litigators, for example, this author was consistently told that 'patents are for the big boys',[52] a statement which did not mean that gaining patent protection was not possible for smaller enterprises, but simply that, if one wished to enforce the right through litigation, then one had better have quite substantial resources or other means of funding.[53] Patents – whatever their actual strength – have historically been used as sticks by competitors, as Henry Ford suggested:

> I believed that my engine had nothing whatsoever in common with [the Selden Patent]. The powerful combination of manufacturers who called themselves the 'licensed manufacturers' ... brought suit against us as soon as we began to be a factor in motor production. The suit dragged on. It was intended to scare us out of business.[54]

[52] This actual phrase was repeated to the author several times by litigators.
[53] The author found little evidence of other funding, such as insurance schemes, in practice.
[54] H. Ford and S. Crowther, *My Life and Work* (Whitefish, MT: Kessinger Publishing, 2003), pp. 62–3.

The fear is that this situation would continue with software patents, and would be heightened by the very large disparity in size between the major and minor software producers. This criticism is common: Haberman and Hill[55] have suggested that part of the problem for SMEs is that there has been a 'system failure' for patent enforcement for SMEs and lone inventors. London shares with the US a relatively expensive litigation system, where costs can rise due to discovery and other procedural mechanisms, and it is also the case that the losing party is held responsible for the costs of the winning party.[56] The Patents Court in London was certainly at the forefront of procedural changes in the 1990s to reduce these costs and speed up litigation; a further advance was the introduction of a new Patents County Court for lower level cases. However, costs of defeat still remain high and any smaller software house would surely be reticent about undertaking to enforce rights against a large software producer in the UK courts for sensible business reasons alone. Germany has lower costs due to a much more restricted procedural system, so it may be that the London perspective is simply reflecting a provincial problem.[57]

Hinder or invigorate?

This first policy argument listed above is important because it opposes the ongoing rhetoric of the EC – that is, that IPRs are always good and more IPRs are even better.[58] There is certainly some evidence which has been put forward that some disruption of the software industry is to be found: in an early objection to software patenting, Garfinkel, Stallman and Kapor[59] used the example of the XyWrite word processing program, which fell foul of a patent granted in 1988 to XyQuest,[60] to argue that the main effect of software patenting was to hinder innovation rather than

[55] M. Haberman and R. Hill, 'Patent Enforcement for SMEs and Lone Inventors: a system failure' (2003), available online at http://www.intellectual-property.gov.uk/ipac/pdf/inventors.pdf. Both are inventors and owners of SMEs and members of the UK Intellectual Property Advisory Committee.

[56] A useful Law Society introduction designed for UK solicitors, *Contentious Costs Practice Advice Service*, is available online at http://www.costs.lawsociety.org.uk.

[57] The Munich patent court appears to be becoming a favourable location for patent litigation. See the Annual Reports with figures of the BundesPatentGericht at http://www.bpatg.de.

[58] A critical approach to European developments is evidenced, for example, in P. B. Hugenholtz, 'Copyright and Freedom of Expression In Europe', in R. C. Dreyfuss, H. First and D. L. Zimmerman (eds.), *Innovation Policy in an Information Age* (Oxford: Oxford University Press, 2000).

[59] S. L. Garfinkel, R. M. Stallman and M. Kapor, 'Why Patents are Bad for Software', *Issues in Science and Technology* (Fall 1991), pp. 50–5.

[60] US 4,777,596.

to invigorate it. A number of other studies have been undertaken since this paper but – overall – there is no clear evidence pointing either way, that such patents invigorate or hinder innovation.

An example of a result suggesting potential (rather than actual) hindering is Wagner's study of franking device manufacturers.[61] Wagner looked at a corpus of European patents which are particularly interesting, given that they relate to business methods: the clear evidence is that few such business method applications are refused by the EPO, though statistically more of these granted patents are opposed than the average patent – 16 per cent of business methods are opposed against 6 per cent of all patents. The outcome of opposition is that 41 per cent of business method patents are declared invalid after opposition (36 per cent for all patents opposed). Franking device manufacturers are few in number (five), and Wagner demonstrates that each has a different patenting strategy, providing negative conclusions as to the value of such patenting activity with respect to smaller concerns. He argues that looking at a micro-patenting area such as this supports concerns about business method patents: that Pitney Bowes:

> relies heavily on business method patents in order to construct a large patent portfolio which is used as bargaining chip in licensing negotiations. This behaviour induces his competitors to fight back by opposing against its patents at an enormous frequency ... The findings from this case study therefore support the concerns that the granting of business method patents might lead to (inefficient) high litigation cost.

On the other hand, Merges's recent article[62] on the effect of software patents on developing firms investigated fifty venture-backed software firms and attempted to ascertain objective measures as to the amount of effort put into (and thus value placed upon) their patent portfolios. In particular, the metrics being used by Merges were trying to determine the quality of the patents being sought – rather than just the number, or patents which were litigationally problematic. Thus, prior art searching before filing was seen as a useful indication of the effort being applied. Merges's findings suggest that successful firms put significant effort into their patents, from which he argues that 'it is safe to say that the predictions of the software patent doubters in the early 1990s [that small software firms would be killed off] have been effectively refuted so far.'

[61] S. Wagner, 'Business Method Patents in Europe and their Strategic Use – Evidence from Franking Device Manufacturers' (2006), available online at http://epub.ub.uni-muenchen.de/archive/00001265/.

[62] R. P. Merges, *Patents, Entry and Growth in the Software Industry* (2006), working paper available at online SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=926204.

Contrary to Merges, Bessen and Maskin[63] have claimed that there is a more 'sequential' kind of invention found in software, where developers rely much more than in other areas upon prior work, and that such a sequential model is better served (and the public better served, too) by having less protection. The authors use other empirical work – Bessen and Hunt[64] – to suggest that when patent protection was available in the 1980s and 1990s, R&D expenditure on software fell in comparison to sales. The argument is that allowing imitation is more productive than providing protection:

> We maintain, furthermore, that there is nothing paradoxical about this outcome. For industries like software or computers, theory suggests that imitation may *promote* innovation and that strong patents (long-lived patents of broad scope) might actually *inhibit* it. Society and even the innovating firms themselves could well be served if intellectual property protection were more limited in such industries. Moreover, these firms might genuinely *welcome* competition and the prospect of being imitated.

A relatively even-handed study by Tang, Adams and Paré for the European Commission[65] in 2001 paid particular attention to the views of SMEs but the authors appeared to find the responses dispiriting,[66] suggesting – as is often the case – that education of the SMEs in the operation and the advantages of the patent system was required. Another European-funded project in 2002, but this time for the European Parliament by Bakels and Hugenholtz[67] on the proposed Computer-Implemented Invention Directive, similarly did not find that SMEs were particularly seeking more protection for software, arguing that proponents were too ready to suggest that patents are good for SMEs, whereas they can – 'in fact', the authors suggest – have serious consequences for these smaller enterprises. In many ways, this echoes the 'common sense' view reported by my interviewed UK IP litigators who were not simply discussing software patents but all kinds of patents.

[63] J. Bessen and E. Maskin, 'Sequential Innovation, Patents, and Imitation', November 1999, revised March 2006, p. 2, available online at http://www.researchoninnovation.org/online.htm#ip2.

[64] J. Bessen and R. Hunt, 'An Empirical Look at Software Patents', Federal Reserve Bank of Philadelphia Working Paper 03–17 (2004).

[65] P. Tang, J. Adams, and D. Paré *Patent Protection of Computer Programmes [sic]*, Final Report, Brussels-Luxembourg (2001). Contract INNO-99–04.

[66] A 'bleak view' as they describe it, p. 73.

[67] R. Bakels and P. B. Hugenholtz, 'The Patentability of Computer Programs: Discussion of European-level Legislation in the Field of Patents for Software', JUI 107 EN (2002).

There have been some studies of the Danish ICT market, which is a classic SME marketplace,[68] with the economists Kaiser and Ronde reflecting on this information and pointing to a number of negatives relating to software patents but then suggest that patent rights 'will stimulate innovation in the software-industry world-wide. This will lead to additional innovations from which Denmark will also benefit. We find it questionable, however, whether this is enough to outweigh the negative effects.'[69]

Litigation of patents is also a factor in assessing their hindrance. According to Allison *et al.*, one of the key characteristics of US litigated patents is that they tend to be owned by domestic rather than foreign companies,[70] suggesting that litigation is something which is easier to pursue on one's home territory. The number of litigated software-related patents in Europe, however, is relatively small and there is little evidence that litigation necessarily follows the US pattern; but it would be reasonable to assume that small European software firms would, if they decide that litigation must be undertaken, prefer to do it in Europe than in the US.

The reader of the literature on software patents could not arrive at any clear conclusion as to whether there is a positive resultant to SMEs from allowing software to be protected by patent. Some evidence does suggest that there can be advantages, but other work indicates that, for the broad generality of small software firms, or small firm utilising software in its products, there may be more negative than positive effect. Should we be surprised by this? Not really, since – as we discuss below – it is inordinately difficult to prove the beneficial effects of patents for any size of commercial concern excepting those in limited fields (e.g. pharmaceuticals and chemistry[71]). If there is such a dearth of clear benefit to

[68] S. E. Hougaard Jensen, U. Kaiser, N. Malchow-Møller, J. R. Skaksen, and A. Sørensen, 'Denmark and the Information Society: Challenges for Research and Education Policy' (Copenhagen: DJØF Publishing, 2003). See also B. Van Ark, R. Inklaar and R. H. McGuckin, 'Changing Gear – Productivity, ICT and Service Industries: Europe and the United States', paper presented at the ZEW conference, 'Economics of Information and New Technologies', Mannheim, Germany, (2002) available online at http://www.ggdc.net/pub/gd60.pdf.

[69] U. Kaiser and T. Ronde, 'A Danish View on Software-related Patents', Centre for Economic and Business Research Discussion Paper (2005), available online at http://www.cebr.dk/publications%20submenu/discussion%20papers/2004/dp%202004-05.aspx.

[70] J. R. Allison, M. A. Lemley, K. A. Moore, and R. D. Trunkey, 'Valuable Patents', *Geo. L. J. 92 (2004), 435.*

[71] Though even here, we see difficulties from expensive litigation. D. Bucknell in 'The global Lipitor patent scorecard' (available online at http://www.duncanbucknell.com) keeps an up-to-date listing of the Lipitor litigation between Ranbaxy and Pfizer as it spreads around the world. For an overview of research and commercialisation of research in chemistry see Chemical Sciences Roundtable, *Assessing the Value of Research in the*

commercial firms, then we can understand why the open source community is even less welcoming to the expansion of protection for software.

Software and examination

This is a problem which follows on from the earlier chapters in this text and is also looked at in Chapter 4, and so will be mentioned only briefly here. A good review of one factor – citing prior art in examination – is highlighted by Merges in his 'As Many as Six Impossible Patents Before Breakfast',[72] albeit examination under the US regime. Merges's argument is that there is a problem with examination – why, he asks, is patent quality so poor? He notes that there is a very limited number of prior art citations in computer-based business method patents (fewer than five) with three of these being to other US patents and also notes the disturbing fact that only recently has patent documentation become available for these kinds of patents, and thus: 'Consequently, we would expect that most of the prior art in this field would be of the non-patent variety. There is every reason to believe that there is a vast volume of non-patent prior art in the software-implemented business concept field, as is widely believed to be the case with software patents in general.'[73] Merges's article is useful because he suggests that, even if examination is problematical, there is an opportunity to resolve this through a 'European approach' – that is, by using opposition after grant, under a scheme he calls 'efficient private party validity-review'.

Merges's ideas have been taken on board both in proposals for law amendment by the legislature[74] and also by various private parties who want to improve examination, such as the Community Patent Project[75] at New York Law School.

Would such a change in strategy improve patent examination? Perhaps; but there are also problems with examination in Europe which have not

Chemical Sciences Report of a Workshop (Washington: National Academy Press, 1999) This text also includes an overview of IBM's software research and development (J. M. Jasinski, 'Assessing the Value of Research at IBM').

[72] R. P. Merges, 'As Many as Six Impossible Patents Before Breakfast: Property Rights for Business Concepts and Patent System Reform', *Berkeley Tech. L. J.* 14 (1996), 577–615.

[73] P. 589. [74] Patent Reform Act 2005, HR2795.

[75] 'The patent system needs our help. The United States Patent Office is actively seeking ways to bring greater expertise to bear on the review of patent applications and ensure that only worthwhile inventions receive the patent monopoly. Currently, underpaid and overwhelmed examiners struggle under the backlog of applications. Under pressure to expedite review, patents for unmerited inventions are approved.' See http://dotank.nyls.edu/communitypatent/about.html.

been solved by having an opposition system. While prior art is certainly central to examination, it is not the only aspect. Of particular importance is the fact that in software we have a virtual, malleable product which can be described in a whole host of ways, a fact which patent attorneys can certainly use in their attempts to gain protection for clients. Even if one has found all relevant prior art, therefore, there may still be problems in determining to what the applicational claims actually refer – it could conceivably be even more difficult than assessing conflict between the springs and rubber slits of the Epilady litigation.[76]

Many of the problems which have been found in the US system can be directly tracked back to the attitude of the CAFC, which has radically altered the patenting landscape in a way which has left the US patent office floundering, both conceptually and with regard to manpower. The pro-patent approach of the CAFC has ignored – some argue[77] – the necessary limitations upon definitions of patentable technology and led to an examination culture which knows no limits. The US debate about patent quality is therefore wider than simply that of software examination and goes to the underlying function of the system as a whole.[78]

The aim of this text is not the same: the author assumes that the patent system will remain – whatever its success in achieving its suggested goals – but that if there are particular problems with software examination then this element should be revised and improved upon: and, as outlined in Chapter 2, there is something about software which potentially makes examination more difficult than in the traditional fields of technology. We look at these issues in Chapter 4.

Suitability for SMEs

The goal of Bangemann is a Europe-wide approach to information economic development and thus the framework is set for a larger economy than the nation state. However, within the EU there are differences between individual states in their IT economy which cause some of these states to fear that disruption or at least more cost is more likely

[76] J. Straus sets the context for this problem in 'The Patent System in the European Union – Status and Development', *Patinnova '97* (1997), available online at http://cordis.europa.eu/patinnova/src/straus.htm.

[77] A. B. Jaffe and J. Lerner, *Innovation and its Discontents* (Princeton, NJ: Princeton University Press, 2004). See Chapter 4 in particular.

[78] See, the wide-ranging analysis in S. A. Merrill, R. C. Levin, and M. B. Myers, (eds.), *A Patent System for the 21st Century: Committee on Intellectual Property Rights in the Knowledge-Based Economy* (Washington, DC: National Academies Press, 2004).

with a software patent protection regime. For example, Kaiser and Ronde[79] put the point that, for Denmark, protection is likely to be more negative than positive, since Denmark is a minor producer of ICT yet a major adopter. This argument is a return to the national perspective of IPRs (which is forever a perspective important to local industry) and requires that members states should give up national benefits for the good of those countries which have larger ICT economies such as the UK and Germany. The further goal of Bangemann is the encouragement of larger firms.[80] It is this which is particularly galling to the European anti-software patent movement, which frequently takes a more 'small is beautiful' approach to economic development, a philosophy expressed by Schumacher when he argued that one's workplace should be dignified and meaningful first and efficient second, and that we need a more collective attitude towards production[81] – the 'open source' movement can be seen as one example of this approach. Stallman, for example, in a 1995 essay talks about the psychosocial harm which affects programmers who are unable to share their work and the technical excitement they feel about this with others, and argues that the greatest scarcity is 'the willingness to cooperate for the public good'.[82]

Bangemann takes a more traditional approach to economics and the labour market and can be read as, effectively, a message to SMEs that the IPR philosophy of the EC is directed towards larger enterprises, and that they should stop being SMEs and grow larger. There are some lessons from history which perhaps support Bangemann more than the open source movement: the history of the machine tool industry in the UK, for example, was one of remaining small and general purpose, a strategy which effectively failed as other countries' firms with larger resources moved into computer control of tooling and specialist tool manufactur-

[79] Kaiser and Ronde 'A Danish View on Software-Related Patents'.

[80] See the Commission's Report, 27 November 2006 from the TASK-FORCE on ICT Sector Competitiveness and ICT. Membership of the taskforce appears to have been primarily large ICT firms, urging – in the author's reading – SMEs to stop complaining and upgrade their business plans and understanding of the new world. Available online at: http//:ec.europa.eu/enterprise/ict/taskforce.htm.

[81] Schumacher was chief economic advisor for the National Coal Board for 20 years, but also a student of Buddhist and Taoist thought. See E. F. Schumacher, *Small Is Beautiful: Economics As If People Mattered* (New York: Harper and Row, 1973). For an example of this approach, see N. Sawhney, 'Cooperative Innovation in the Commons: Rethinking Distributed Collaboration and Intellectual Property for Sustainable Design Innovation', PhD dissertation, MIT (2003).

[82] R. Stallman, 'Why Software Should be Free' in D. G. Johnson and H. Nissenbaum (eds.) *Computers, Ethics and Social Values* (Englewood Cliffs, NJ: Prentice Hall, 1995).

ing.[83] Such a message is politically controversial and this, therefore, has been one that the EC has been keen to undermine: that SMEs are just as likely to benefit from the patent system as larger enterprises, even though the evidence is perhaps slightly more towards SMEs being hindered as Kingston suggests, when '[s]ome cases were found where firms did get some compensation from their infringers, but in far more cases the evidence from this research was that *patenting* does not pay SMEs'.[84]

Opposing this view, there have been arguments posited which suggest that patent rights are the *only* suitable method whereby small software innovators can protect themselves from large aggressive plagiarists and, certainly, given sufficient access to resources to litigate, the smaller innovator will then be on equal terms with the larger enterprise. Insurance schemes have been proposed as one method of levelling the litigation playing field:

> As things are today, there are presumably a number of companies which consciously calculate that they can avoid prosecution for patent violations. A European patent insurance scheme would function like a burglar alarm. If the intellectual property thieves know that they will be prosecuted for infringing competitors' patents, they are more likely to resist the temptation.[85]

A similar idea, a Patent Defence Union (PDU), has also been proposed by Kingston as a method of protecting the SME, through effectively forming a group which will protect each other and would also encourage non-litigational methods of resolving difficulties. He has suggested:

> It will be evident, then, that the very existence of the PDU could swing the balance of the odds back strongly towards SMEs, by making intimidation a less effective weapon for large firms. It would change the environment for decision-making by managers in such a firm in terms of their career paths so as make them cautious about infringing.[86]

This social engineering of management's caution would arise, Kingston argues, because, if their case was strong, they would agree to arbitration.

[83] See R. Lloyd-Jones and M. J. Lewis, *Alfred Herbert Ltd and the British Machine Tool Industry, 1887–1983* (Aldershot: Ashgate, 2006). On the other hand, it has been possible for some firms to remain small and successful. Peter Stubs Ltd continues (though no longer in private hands) as a small specialist firm long after its file-making began – see E. S. Dane, *Peter Stubs and the Lancashire Hand Tool Industry* (Altrincham: John Sherratt & Son, 1973).

[84] W. Kingston, *Enforcing Small Firms' Patent Rights* (Dublin: University of Dublin Press, 2000), NB-NA-17-032-EN-C. Kingston's other work is interesting too: see, e.g., 'Limited Incontestability for small firms', *E.I.P.R.* 463 (2006), where he suggests that the patent system is biased towards use by large firms.

[85] Henrik Dahl Sørensen, Deputy Director General of the Danish Patent and Trademark Office, Danish Ministry of Foreign Affairs, 24 October 2002.

[86] Kingston *Enforcing Small Firms' Patent Rights*.

If weak, they would negotiate a licence, since infringement carries a risk of litigation by the PDU in defence of its members. A different tack has also been proposed suggesting that *utility model* protection would be a suitable form for SMEs – requiring a lower level of invention in return for shorter and more local protection. We look at this in Chapter 6.

Monopoly issues

There are – opponents suggest – valid reasons for the opposition to software patenting because of the potential monopoly problems where 'standards' which software developers who wish to integrate with other software or communication formats are required to use. For example, the LZW patent[87] at the core of GIF compression is an example. The algorithm which carried out the compression technique was utilised in several standards – including CompuServe GIF file format – despite the fact that it had been patented. Some suggest it was due to having been discussed in the June 1984 issue of *IEEE Computer* and there being an assumption that it was therefore public domain knowledge. This file format gives relatively 'lossless' compression and is one used most frequently on web pages, being very successful in compression and also enabling moving images. Unisys, the owners of the patent, to the chagrin of web developers, decided in 1994 that it required royalties from commercial developers who included GIF creation facilities in their programs. In 1999, they extended this to all – not only commercial – programs. To the anti-software patent movement, this was a prime example of the dangers of software patents.

This is effectively similar to a 'submarine patent'[88] – waiting under the surface to strike only when and if the patent is worked. In software, it is usually possible to work around such a patent – as was done with the PNG

[87] US 4,558,302, 'High speed data compression and decompression apparatus and method'. Richard Stallman actually suggests that there were two patents covering the same algorithm: 'If something is purely mathematical, there are many ways of describing it, which are a lot more different. They are not superficially similar. You have to really understand them to see they are talking about the same thing. The patent office doesn't have time. The US Patent Office as of a few years ago, was spending on average 17 hours per patent. This is not long enough to think carefully about them, so, of course they make mistakes like that ... That algorithm also had two patents issued for it in the US. Apparently, it is not that unusual.' Transcript of a talk, 'The Danger of Software Patents', presented on 25 March 2002 at the University of Cambridge. Available online at http://www.ftc.gov/os/comments/intelpropertycomments/stallmanrichard.pdf. Unisys has a current application in the compression field (EP1453207) so has not, seemingly, lost its desire to patent this technology.

[88] Of course, formally, a submarine patent is one which has not been granted and lies – under the US system – unpublished until grant. Publication has now made these an endangered species.

and MNG compression formats – but by that time JPG was in effect the standard image format on the web. Unysis were in a strong position to collect a revenue stream from many users of the standard – a relatively low licence fee would bring in large sums of money, whereas a higher fee would drive users to develop workarounds or take up the alternative approaches. For many commercial developers, the licensing costs appeared to be acceptable, but for the open source community, where there is no or little income, the costs were more important.[89] It has been suggested that the open source operating system, Linux, could potentially have accidentally implemented some 300 patented ideas – a point that Microsoft may not have been adverse to highlighting.[90] Does this matter? Could it potentially undermine the growing use of Linux? Not according to Linux creator Linus Torvalds, who has been reported as not being worried by suggestions that Linux may infringe on patents:

> Hey, there 'may' be life on Mars. What does 'may' mean?" he says via e-mail, adding that if Linux really does infringe on a patent, he'll just rewrite the code to sidestep the problem.[91]

What is surprising, perhaps, is how few of these problems appear to exist. We have had many years of protection for software ideas and yet the most prominent remains LZW and most other problems for the open source community are 'potential' rather than 'actual'. This is not to say that patent infringement cases are not being heard,[92] only that the worst-case scenarios as described by the open source community have not arrived. The evidence is that the software community may well have learned from LZW and that it is much more careful about utilising approaches as standards which may be proprietary – and an effective and politicised open source community has been keen to ensure that earlier mistakes are not repeated.[93] Standards have been potentially problematical for many

[89] For a short review of firms' strategies relating to open source standards, see S. Weber and D. Lancashire, 'Open Source and Standards', Berkeley Roundtable on the International Economy (BRIE) (2004), available online at http://programs.ssrc.org/itic/publications/ITST_materials/weberchapter.pdf.

[90] M. J. Foley, 'Is Microsoft Rattling the Linux-Patent Sabers?', *eWeek.com* (18 November 2004).

[91] D. Lyons, 'Linux Scare Tactics', *Forbes.com* (February 2004). The article also refers to the open source community 'scaring itself' by offering insurance against patent litigation.

[92] The classic example being the BlackBerry litigation, though the patents here are, at the time of writing, under re-examination and appear to be weaker than thought.

[93] 'W3C's evolving patent policy has been informed by help, comments, criticism, and occasional rants by W3C Members, many voices from the independent developer and Open Source/Free Software communities, W3C Advisory Committee Representatives, the W3C Team, the W3C Advisory Board, and participants in the Patent Policy Working Group.' See http://www.w3.org.

technical fields and standards bodies have developed means to reduce the potential conflict between open usage and technical standard which processes have been reasonably well discussed in judgments. Lemley has described the substantive law and the formal process of agreeing standards and the requirements which members of standards organisation undergo in order to participate.[94] This process has become central to the working of the W3C patent working group,[95] for example, with a clear policy agenda of only including proprietary technologies when it is available on a royalty-free licensing agreement.[96]

Monopoly issues are certain to grow in software – for example, the European Commission and Microsoft over access to interface information[97] – but it is not clear whether the existence of software patents *per se* makes the situation very much worse. There is more of an understanding of the potential for conflict – raised very effectively by the open source community – that core standards at least will be available to developers. Non-core standards – those in effect due to the 'network effect', where users congregate and make a *de facto* standard, are more difficult. These are rarely standards which evidence the very best technology and can be worked around, but – as IBM's PC rival operating system to MS Windows[98] found out – a *de facto* standard can be very difficult to remove. Such standards can be developed without use of patents, of course: Gifford, in his analysis of standard setting models has pointed to the fact that once network effects are in place, competitors find it very difficult indeed to unseat the beneficiary of that effect,[99] which must mean that better technology – whether protected or not – is not always successful in the marketplace.

[94] M. A. Lemley, 'Intellectual Property Rights and Standard-Setting Organizations', *Calif. L. Rev.* 90 (2002), 1889.

[95] The World Wide Web Consortium is responsible for development of web standards.

[96] E.g.: 'With respect to a Recommendation developed under this policy, a W3C Royalty-Free license shall mean a non-assignable, non-sublicensable license to make, have made, use, sell, have sold, offer to sell, import, and distribute and dispose of implementations of the Recommendation that: *1. shall be available to all, worldwide, whether or not they are W3C Members*; . . .' W3C Patent Policy, 5 February 2004 (emphasis added).

[97] Commission Decision of 24.03.2004 relating to a proceeding under Article 82 of the EC Treaty (Case COMP/C-3/37.792 Microsoft) C(2004)900 final.

[98] OS/2 was viewed by many as technically superior to MS Windows. However, it never took off commercially and IBM announced end of sales in 2005 and end of support in 2006. It will remain in modified form from another vender as eComStation. IBM considered making it open source, but some of the code is from its early collaboration with Microsoft and thus agreement to this is not likely.

[99] D. J. Gifford, 'A Developing Model for a coherent treatment of standard-setting issues under the patent, copyright and antitrust laws', *Idea* 43(3) (2003), 331–94.

Do 'workarounds' weaken software patent strength?

The strength of a patent in part comes from the inability of competitors to 'work around' the patent – that is, the protected technological solution to the problem is, for example, the cheapest, the best or the only one available. Thus, in pharmaceutical patents, we can see that the relevant reason for strength of a patent is that the successful compound is the only one available. In physical fields, this is rarely the case and it is usually possible to work around the patent[100] given time and effort to redesign the product and the processes, and thus patent owners will try to develop a protective screen around their core patents to provide protection to competitors' 'workarounds'. They will also – using the PCT route perhaps – hold off the final text of their patent and claims until they know the direction in which the competition is moving.[101] Competitors must balance the economics between licensing or finding alternative approaches which work around the patent, etc.

The 'workaround' is one of the many common maintenance procedures in computing: changes to every program are needed, older coding must be made to fit new requirements, and the role of the maintenance programmer is primarily carrying out these 'temporary' improvements which – in all likelihood – will be permanent. Fixing security holes is another example of this kind of task. Programming is ideal for achieving workarounds – there is no need for replacement or alteration of manufacturing process and the inherent malleability of software means that alternatives can be quickly put into place. Thus, even if a very speedy algorithm is protected, it will be possible to replace this with one which is non-protected – it may be half the speed of the other, but processing tasks are rarely dependent upon the speed of one operation, and so the overall speed reduction may not damage the competing product's value to the consumer if it has other positive aspects (cost, user friendliness, etc.). A protected 'one click' purchasing option can be replaced by a 'two click' option or some such.[102] Linus Torvalds's approach (mentioned above) to perceived problems with patented technology in Linux is another example (though this is not to suggest he is a pro-software patent proponent). Moreover, these replacements can be done – in comparison with other

[100] This does not apply when standards are involved.

[101] In an earlier study, this was a relatively commonly proposed advantage by patent attorneys. See Leith *Harmonisation of Intellectual Property in Europe*.

[102] As Barnes and Noble did under threat of Amazon.com's patent US 5,960,411, 'Method and system for placing a purchase order via a communications network'.

technologies – in a very speedy manner and can be transmitted to users quickly and speedily.[103]

It is possible that the ease and flexibility of producing workarounds effectively undermine many of the advantages of holding software patents at all. This depends upon what has been allowed for protection: if one possible solution to a problem is protected, then it is likely that there are many other alternatives; if protection has been given for the problem itself (for, say, 'the use of a computer to assemble documents') then workarounds are clearly of much less utility to competitors. Also, if there is a lack of clarity in the claims about just what is being protected, that, too, affects the ability of competitors to be confident that their workarounds do not infringe.

The aim of the patent system is both to protect and to encourage, and a well-carried-out examination should – we suggest – be able to provide protection and sufficient room in which to produce other solutions to the same technical problem. This is an area where practice and policy clearly merge.

Conclusion: the patent system as lottery?

In some arguments supporting the patent system, there is a suggestion that the underlying aim of the system is directed towards socially worthwhile results. This is usually put in terms of a short monopoly being unwanted but worth accepting for advancing 'social benefit'. As worthy an aim as this is, the examiner cannot, of course, take into account such factors as the social value of a patent application: applications from one person which relate to patent value assessment software (US 6,566,992) must be examined in the same way as those which relate to magic wands and interactive play experiences (application EP1606031). Given that it is difficult to predict whether any given patent will have any economic value at all, let alone social value, the role of the examiner would be untenable if they were required to make a subjective decision on such social value. Arguments which rely on social benefit as a justification for the system or as a means to critique it are therefore really only usable at

[103] In fact the workaround in computing is such a successful method to undermine anti-circumvention copy protection that legislation has been seen to be required: the copy protection routines being easily overcome by other programming methods. Of course, one of the dangers of this methodology is that it creates 'forks' in system development which results in programs which should be similar or identical in failing to interact but it is also used as a strategy by the major software developers who deliberately accept standards and then move the standards into an essentially proprietary format through forking to upset the competition.

the macro level. At the micro level of examining patents, the examiner follows the procedure outlined in the examiners' guidelines, yet if these guidelines allow patent grant for inventions of little social worth, then the larger whole (which may indeed be valuable) loses perceived value.

Having read this chapter so far, the reader will surely be in no better position to decide whether software patents are beneficial at the macro level or not: the evidence is simply not striking enough. The patent system as a whole has been the target of research by economists for many years but, much like the more recent attempts to locate either benefit or disadvantage from software patents, it has been difficult to arrive at clear conclusions as to the system as a whole. Economists generally are not welcoming to patent monopolies because they view monopolies as inherently a 'market failure' and suspect that they encourage 'rent seeking' rather than competition. Yet despite this underlying critical perspective, it has certainly not been possible to prove that patents are totally disruptive, even though, likewise it has not been possible to prove that they are totally supportive and rational instruments of economic policy. For example, Hoppen's simulation models of the various software patenting environmental factors which might affect the systemic value of these protections is certainly interesting but cannot, as one suspects he would agree, yet provide an answer. His work, however, highlights issues which are still unresolved in our economic study of the system.[104]

Great rewards are sometimes available through the patent system: at present in the US some of the awards made in patent litigation appear spectacular and out of all proportion to the infringing act – an act which was probably done without thought and which, given the flexibility and malleability of software, could perhaps have been 'worked around' without too much effort. But many of those who are granted patent protection will not see such reward, and the majority will lose protection through deciding not to pay renewal fees. The commonly held anecdotal view is that one per cent of patents are profitable, 10 per cent cover their costs, and the remainder are of no or little economic value at all.

The critics of software patents have looked at this in a rational light and suggested that, given that the vast majority will not benefit, the system fails – the argument being that if the majority do not 'win' – and may even lose – then there is little point in having a system which it is so difficult to prove is supportive of innovation. Are they correct? Is patent policy irrational? Perhaps, but there is an intuitive argument which has been put by F. M. Scherer, based upon the supposition made by Schumpter quoted at

[104] N. Hoppen, *Software Innovations and Patents: A Simulation Approach* (Stuttgart: Ibidem-Verlag, 2005).

the beginning of this chapter. Scherer, a researcher in the field of economics and IP for many years, has suggested that there may indeed be an underlying principle which explains why the system works and yet appears to be irrational. His premise is that if we accept that there is a skewness which mirrors that found in other areas such as the sales of records (Led Zeppelin's sales comprising 17.6 per cent of the top forty-eight artists, gradually tailing off as the artists are further down the list and then dropping off the scale for the vast majority of artists), then we begin to suspect that there is a principle which might be underlying the patent system: that of the *lottery* – many enter but few win. Great rewards will go to some, and the carrot of those rewards is sufficient to ensure that large numbers of individuals will spend time, money and effort on attempting to achieve these great rewards which are only available to the few. It is not essential that everyone enters a lottery, but for those who are inspired by winning (not, we suppose, the open source community) they can be egged on to spend, despite the statistical fact that they are unlikely to win – the anecdotal one per cent of patent grants simply showing *a* profit, rather than the embarrassingly large profits which must be available to an even smaller percentage.

Thus, large sums can be spent on R&D by many firms, yet only the first to file[105] will be rewarded. Without the potential to win the top prizes, perhaps there would be much less pressure to do R&D at all – even if that R&D is directed towards those areas which offer the biggest prizes (equivalent to, in the cultural context, popular music rather than a new composition by the *avant garde*). If Scherer's argument has substance, what does this mean for intellectual property's claim to offer 'balance' to all the users of the system? A lottery, we might suggest, is hardly a balanced affair. A lottery, though, requires clear rules to determine the winner and perhaps that is what the examination system really is, rather than an attempt to balance the various demands of patent breadth and scope.

Should we be disturbed by suggestions that the system is lottery-like? Some legal theorists believe that the idea of incorporating chance within decision-making is not as radical as we might think, and nor should it be totally ignored as an appropriate method for dispersing goods. For example, Duxbury[106] has pointed that there is a considerable history – showing both advantage and drawback from using such seemingly random legal

[105] This is in Europe, of course. The 'first to file' rule is politically sensitive in the US – attempts to change this have been unsuccessful – where smaller concerns argue that this simply advantages those large concerns who can do R&D quickly.

[106] N. Duxbury, *Random Justice: On Lotteries and Legal Decision Making* (Oxford: Oxford University Press, 2002).

decision-making – but that sometimes the most 'cost-efficient and impartial decision-making strategy may well be recourse to lot'. Aversion to this possibility, he suggests, shows a perspective where reason and process are more highly valued than outcome. Of course, the patent lottery system (if that is what it is) is one where the marketplace would interact with the legal system to produce an efficient and effective outcome, with only minimal regulation of process.

The idea of the system having lottery aspects is intuitively appealing to those who have investigated the workings of the patent system – the tactical and strategic use of patents melded with the developments in particular technologies appear to have a very large element of chance and yet these aspects do not appear to very great extent in the law books or patent decisions and judgments. Some may suggest that this is a nihilistic perspective of the system, but it need not be: examination may be more than just the setting of sweepstake rules, and may inject a sufficient measure of rationality into a market system which would otherwise be one of bigger dog eat smaller dog.

4 Software patent examination

When I have a new trainee, on his first day I tell him . . . two things. The first is you do exactly the same when writing a story – you collect the facts and you must give them a structure, the structure of the claims. And if you are a good author, then that is the first requirement. The technical element you can learn from the person who comes to you – there is no need to know anything about the topic you are working on. But then when preparing a case you have to convince the examiner of the merits of the invention. And that is not a point of technical ability but of psychology.[1]

Introduction

Whilst amongst a group of postgraduate IP students, I made a statement which I thought was relatively non-controversial: that getting patent protection was not too difficult. The law students, almost as one, let out a groan of dispute which clearly suggested that they considered me wrong. Is this a view that most lawyers hold? Perhaps more comfortable with the substantive law of patenting rather than the technology being discussed – noting the seeming highly technical nature of the patent specifications and concluding that it must 'all be very difficult' indeed? It is not clear why the students viewed patent examination as such a significant hurdle to overcome. The EPO figures demonstrate the truth of the statement: the 2005 Annual Report notes that for 128,679 Euro and Euro-PCT applications, the number of patents granted was 53,259, which represents an applicant with a 41 per cent chance of success, which is hardly an indication of great applicational failure, given that this is a monopoly which will last for twenty years.[2]

[1] Quoted in P. Leith, *Harmonisation of Intellectual Property in Europe: A Case Study in Patent Procedure* (London: Sweet and Maxwell, 1998), p. 50.

[2] In the US for 2005, there were 390,733 applications and for that same year 143,806 grants made. This is a 36.8 per cent success rate. See http://www.uspto.gov/go/taf/us_stat.htm for fuller details of application and grant since 1963.

Presuming a good quality examination, there are several reasons which might explain this success rate. It may be that many applications are being made by researchers who are well aware of the prior art[3] and what other companies are achieving, and thus will primarily choose those ideas which are indeed novel and inventive. Good patent attorneys will also know how to format an idea into its many appropriate applicational forms, and will – wherever possible – get at least some protection for the idea, if not all that was originally hoped for. Further, many applications would be of the small, incremental advance nature rather than fundamental advances in technology.

On the examination side, a high success rate might mean that the examiners were doing their job well and that the applications were of good standard. On the other hand, it might indicate that the examiner had set a relatively low hurdle of inventive step, had poor access to prior art, or was prepared to give the applicant some undeserved measure of protection by reduced or amended claims. It is very difficult to decide in absolute terms whether Europe has a 'good' examination or a 'poor' examination, simply owing to the large numbers of applications involved, the nature of the examination process, and the diverse views of what is 'good'.[4] One approach in attempting an insight into the process is to look to patent attorneys' views,[5] but while this is certainly an indication and does give rise to a wide variety of views, the patent attorney is principally targeting their client's satisfaction[6] with a particular application and therefore, even though a general lower hurdle of inventive step may damage a particular client in the long term, it is better for the attorney in the short term. Another way might be to look at how many patents are opposed after grant. The 2005 EPO Annual Report puts the figure at 2,960, which means that 5.6 per cent were opposed – a reasonably low figure. Does this mean that, generally, those not on the receiving end of grant were happy with the general standard? Not necessarily – opposition is expensive[7] and companies would be expected to choose which patents they opposed and would certainly choose only those which were potentially obstructive to them, and they may well choose to oppose a patent

[3] The author's earlier research suggests that many applicants do not do a search of patent literature prior to filing, and neither do many patent attorneys. See Leith, *Harmonisation of Intellectual Property in Europe*. The general view from research is that patent literature – despite the encouragement of patent offices – is a rarely used resource.

[4] Agreement is easier over whether prior art is being found, but not over levels of inventive step.

[5] Leith, *Harmonisation of Intellectual Property in Europe*. [6] *Ibid*., pp. 63–4.

[7] Anecdotal evidence seems to suggest £20–£30,000 in total for a serious opposition. The EPO fees are low, but professional fees are high and prior art searching is usually required.

they feel is of good quality, simply in the hope that some prior art arises or the Office will reduce the breadth of protection or amend claims, etc. Opposition is a cheaper option than litigation and it makes commercially better sense to oppose early than to litigate late, though it may make more commercial sense to do neither.

Whatever the situation, it is clear that good examination is the goal which the system should aim to achieve: it does not give unwarranted monopolies, it reduces the overheads of successful oppositions and greatly removes the unwanted litigation which results from over-broad or invalid patents. Unfortunately, good examination is expensive to achieve in all technical fields.[8]

One of the underlying themes of this text is that software has potential difficulties in description and thus in examination. In Chapter 1, it was argued that the patent system had approached this problem by restating software inventions as machine-like inventions. In Chapter 2, it was suggested that a software approach to examination could be taken, but that it required a change of vision of examination: that is, seeing software as software as such.

A number of problems arise which are particularly relevant to software examination and which the system would have to overcome before we could really describe examination as 'software focused'. In this chapter, several relatively simple examples will be used to demonstrate in practice some of these difficulties. They arise from the fact that technology is not brute-fact-like, and that a great deal of interpretative reasoning must occur if we are to compare one invention with another, and see whether the first is really prior art or the second is inventive over the first. Like legal reasoning, technical reasoning is interpretative.

The legal academy is directed towards substantive law problems when analysing patent cases, seeking to extract legal rationales from various judgments on technical matters. The problem with that approach is that, in patent matters, it is frequently the information which is omitted from the constructed legal rationale which is actually the important element. To put it another way, the construction of facts is the most difficult part of the exercise and, once we have decided upon our facts, then explicating the patent law follows reasonably easily afterwards given that the facts and law are interconnected. Thus, discussions of obviousness and inventive step are of legal interest, but it is not possible to provide a separate legal

[8] Some believe that psychology can come to the rescue, but I personally would suggest technical expertise – see C. Dent, 'Decision-Making and Quality in the Patent Examination Process: An Australian Exploration', Intellectual Property Research Institute of Australia, Working Paper No. 01.06 (2006).

mechanism to decide obviousness or inventive step (and even novelty can be difficult), since these can only be decided with reference to the internal reasoning of a technical expert, and such is a matter dealt with under the rubric of 'fact finding'. This means that, often, legal writings on software and patenting seem – to this author at least – to miss out on the more subtle aspects.

Given that few lawyers will have a very high level of understanding of software technology, examples are used in this chapter which are relatively easy for lawyers to comprehend and yet cover some of the many controversial areas in examination. They focus first upon three elements which critics of software patents have seen as important:

1. The apparently wide scope of some software patent claims and the resultant difficulty of deciding just what they protect.
2. The sometimes-poor quality of prior art searching and its effect upon the examination process.
3. When is 'It's just a mental method computerised' relevant?

The examples chosen are not necessarily favourable to a pro-patent argument – particularly the first two, which might be viewed as examples of faults in the patenting system rather than the ideal. But those who see software patents as necessary or inescapable should be expected to propose methods of overcoming faults or mitigating their effects. The thrust of the argument which follows is that – as with any technology – there are specific areas which are problematic to examine and, the more novel the technological field, the more inconvenient we must expect these areas to be. This certainly does not mean that examination is impossible, but it does imply that there will be divergences in practice between the various offices and the courts (should there be litigation or appeal) because there is no simple method for totally reducing human interpretation to some mechanistic, algorithmic procedure.

Also relevant to examination are the common problems which are found in all technologies. For example, the examiner could (by giving too wide a scope) give protection to the problem, thereby denying others the ability to protect alternative solutions to that problem. For example, protection may be given to 'control of robotic arms' rather than simply to one method of controlling these arms. A further problem which is found in all technologies – but may be more difficult in software – is the evidential matter of deciding whether the 'invention' actually works. Thus, we could imagine a patent application claiming a 'computer system implementing reasoning' but might dispute whether this is 'reasoning' in any commonly accepted form. How might any evidence be considered in support or refutation of this?

Breadth: HyperCard versus Zoomracks

The first of the examples is the 'Zoomracks' patent (or rather the patent which was filed by the developer of this software). In discussion of software patents, one of the most-often-cited articles is that by Heckel, entitled 'Debunking the Software Patent Myths',[9] which is best characterised as a defence of software patents for the smaller software inventor. The theme of the article was that those who criticised software patents misrepresented the reality of the situation – including those critics such as the League for Programming Freedom who had categorised Heckel's patent as an 'absurd patent'.[10] Heckel was the owner of a patent (US 4,736,308) which allegedly covered the technology behind the implementation of HyperCard. This is a US patent rather than a European one, but the point of discussing it is not related to the national procedures, rather because it is a technology which is accessible to lawyers – basically the idea of 'links' between elements on screen displayed 'cards' utilising a 'card and rack' model; and because it highlights specific problems which can be found in what is actually being claimed in a software patent. Heckel had worked at Xerox PARC and was author of a well-received text on user interfaces. His program, Zoomracks, was designed for the Atari (this was prior to the hegemony of the IBM PC clone) but 'it was a financial struggle largely because Atari did poorly'. Heckel notes that Apple's introduction of HyperCard, 'which is based on a similar, but more limited card and stack version of the metaphor', created a situation which could not be met by his own program.[11]

Heckel echoes the cry of the small inventor who believes that he has a patent which is being infringed by a larger concern:

> I was then faced with having invested six years of raising money, developing a product, marketing it, and proving its value in the market, only to find I was in debt, my customer base was on a dying computer and Apple was giving away *free* a more polished and featured, although less elegant, version of the metaphor. While Apple may not have set out to rip off Zoomracks, it was aware of Zoomracks (having seen it under non-disclosure), of HyperCard's similarity to Zoomracks, and that Zoomracks was protected by patents.

[9] P. Heckel, 'Debunking the Software Patent Myths', *Communications of the ACM* (June 1992), available online at http://www-swiss.ai.mit.edu/6805/articles/int-prop/heckel-debunking.html.

[10] 'Absurd patent' has seemingly become the preferred terminology to describe a patent which one feels to be invalid or obvious. It is a term which has the necessary dismissive quality for its task – but does sometimes appear to be an accurate descriptor.

[11] Presumably, he is referring to the 'network effect' as a program becomes a standard around which customers and developers gather.

In a single paragraph, then, we have a forceful advocacy for software patents: they provide a way to protect the inventive effort of the small developer who cannot trust larger concerns to abide by trade secret/confidentiality agreements. Apple agreed licensing terms and, in the words of Heckel, 'are to be applauded for respecting my patents'.[12] Of IBM's response, Heckel is less flattering.

HyperCard

The HyperCard program was one of the first so-called 'killer applications' – programs which were so popular that they drove hardware sales; in this case the early Apple Mac computers. The idea is relatively simple: that information could be stored on cards which were 'hyper-linked' together. Further, though, the user was enabled to program these and it was this which really gave the program a hugely popular response. The programming ('scripting') language used was HyperTalk, a relatively simple language which meant that those who were non-programmers (in the traditional 'third generation language' sense) could produce database and applications programs in a number of areas.[13]

Apple gave the software away for free from late 1987 and seemingly had a non-committal attitude to a program of which they did not appear to understand the implications. However, HyperCard was an innovative product on the marketplace, though there appears to have been an earlier form – NoteCards –developed at Xerox PARC in 1984.[14] Of course, Xerox PARC gave the computing world much which has been forgotten – including the InterLisp programming language used in my own computing research – so it is not a surprise to see another inventive idea which arose from that research institution.

The idea behind HyperCard is that there is a collection of cards (called a 'stack'). Each card can contain similar pieces of information and cards will have a common layout (e.g. background photograph, locations for information). Users can decide what information is contained on the card, and can arrange links between cards – the 'hypertext links' – so

[12] 'People close to the industry said the settlement was a one-time payment and did not involve future royalties. Officials of Apple, based in Cupertino, Calif., did not return telephone calls, and Mr. Heckel declined to comment on the suit.' *New York Times*, 20 December 1989.

[13] It has been suggested that VisualBasic is Microsoft's version of HyperTalk. Much of the power of HyperTalk comes from the large number of native functions so that users had many easy-to-use operations already programmed.

[14] R. Trigg, F. Halasz and T. Moran, 'Notecards in a nutshell', *ACM SIGCHI Bulletin* 18(4) (1986), 45–52.

that one could include a map on a card and use a part of the map as live link and when clicked upon, bring up pictures or facts or other materials on that area of the map. We currently find this an obvious idea: this is an elementary internet browser, but in the pre-internet days of the 1980s, this was radical and inventive.

A very useful visual overview of HyperCard is available, including an interview with the 'inventor' via the Internet Archive,[15] which demonstrates the enthusiasm ('fun' appears to have been the keyword) which was found to surround the program, in part because it allowed those with various interests to produce 'stacks' (hypertext linked documents and images) which gave a dynamic quality to their interests – for example, the teacher of composition could produce a 'stack' which helped students follow scores and which included historical and musicological information. The significant move away from traditional programming models was a major factor of HyperCard's success – the user could develop stacks interactively through adding first one button to a card, seeing how it looked or performed, and then adding more. It was, in many ways, a revolution in programming. The inventor of HyperCard was Bill Atkinson, who later reported having failed to foresee the potentially huge inventive step which HyperCard suggested:

'I have realized over time that I missed the mark with HyperCard,' he said from his studio in Menlo Park, California. 'I grew up in a box-centric culture at Apple. If I'd grown up in a network-centric culture, like Sun, HyperCard might have been the first Web browser. My blind spot at Apple prevented me from making HyperCard the first Web browser.'[16]

A number of groups exist to continue HyperCard development, and there appear to be continuing commercial usages: for example, it has been reported that the lighting system in the concert hall for the Petronas Towers (completed in 1998) in Kuala Lumpur is a HyperCard application.

From where did the original inventive metaphor arise? The model from which HyperCard developed seems to have been one which is well known now as 'hypertext' but which has been argued to have grown from an article by Vannevar Bush in the *Atlantic Monthly* describing research into a way of organising information through a machine called a *Memex* in 1945.[17] His Memex machine was to have a large number of documents

[15] See http://www.archive.org/details/hypercard_2. The recording is KCSM TV's 'The computer Chronicles', 8 January 1990.

[16] http://www.wired.com/news/mac/0,2125,54370,00.html.

[17] V. Bush, 'As We May Think', *Atlantic Monthly* (1945). See P. Leith and A. Hoey, *The Computerised Lawyer*, 2nd edn (London: Springer-Verlag, 1998). See discussion in Chapter 2 of hypertext and legal materials.

(and also photographs, diagrams, etc.) and a means of arranging links between the documents (or other elements) in the store of documents. These links were the important element of the machine, because they allowed the user to put a structure upon the materials in the database of the machine. The structure to each individual user was a particular 'trail' through the database. When users created new trails of links, these became the new structure. As Bush put it:

> It is exactly as though the physical items had been gathered together from widely separated sources and bound together to form a new book. It is more than this, for any item can be joined into numerous trails.

The Memex operation can be modelled by thinking of a large library. No-one would ever be able to comprehend all the materials in that library. However, researchers might well be able to make links between the books in the library and produce descriptions ('trails') of different subject areas. These could then be handed on others to use to help them navigate through the library. Memex was an idea, never an implementation – the hardware of the time could have been forced into a system which implemented it, but the newer screen-based GUI interfaces were really required to take the metaphor and, metaphorically, 'run with it'.

The 'rack and card' patent

Just as HyperCard was viewed as a novel and inventive program by its users, so was Zoomracks, a program which is also based upon a card metaphor:

> The idea behind Zoomracks was (and still is) a good one. So good, in fact, that others – with or without knowledge of Zoomracks – built similar card-based schemes of their own. Frank Halasz, for instance, designed 'NoteCards' at Xerox PARC in 1986. Developed on Xerox Lisp machines using the InterLisp programming environment, NoteCards was a hypertext system based on cards, 'fileboxes,' and the links between them.[18]

> It's a new computer concept that surely must change the way databases are used. I challenge you to find a single program that gives you a more comprehensive way to use your [Atari] ST to keep track of your life.[19]

The deviser of the program, Heckel, filed his application in July 1985 (developed from an earlier abandoned application of September 1984). The main thrust of the invention was a 'zooming metaphor', which

[18] See a short overview of the various alternative models in J. Erickson, 'The Real Deal', *Dr Dobbs Journal* (22 July 2001).
[19] L. Ellingham, 'Review', *New Atari User* 23 (Sep/Oct 1986).

allowed viewing of records on hardware with limited screen display size.[20] The idea behind the patent was that one could – using the screen as a kind of camera zoom lens – zoom into and out of a display of records from one or more files. The records could be viewed in shortened form (where only the relevant fields were displayed) or multiple records could be shown on the same screen. The patent contained a relatively detailed flowchart of a program which allowed this, but the techniques used in the program were well known. The inventive idea, therefore, lay either in the metaphor or the implementation of that metaphor, rather than any new programming. Heckel's patent was primarily described in terms of the underlying implementation but also included an analogy to aid understanding:

> In the particular embodiment discussed below, *the system can be compared to a series of racks such as is seen adjacent to a time clock. Each rack is comparable to a file with each rack or file associated with a division or department. Each record is comparable to an individual time card*. The elements of information on each time card such as name, address, pay rate, etc. are comparable to the data elements or fields. Hereinafter the terms file, record, and field will generally be used in the context set forth above. When referring to the 'rack/card' analogy, reference may be made to a column to signify a 'rack' or file and to row to indicate the positioning of 'card' or record in the file.[21]

It is clear that the focus of the patent was on the implementation rather than the metaphor.[22] In the diagram (Fig. 4.1) we see this implementational structure, where the lowest part of the diagram shows the file or collection of files from which the information is to be presented, the image above shows that a record has been 'zoomed into', and the topmost section of the diagram shows how the user can see either an abstracted version of the record or the detail of that record itself. As the patent states:

> This capability can be compared to a zoom lens on a camera where one gets either a detail look or a broad view. It differs from the well known 'window' in that the instant invention provides the 'magnification' of a zoom lens while a 'window' overlays a different portion of the detailed view without the 'magnification'.

The twenty-three claims of the patent were concerned with the physical implementation of this zooming structure, rather than claiming the zooming metaphor itself.

[20] In terms of the amount of information computers can hold, it can be argued that all machines have a limited screen display.

[21] US 4,736,308 (emphasis added).

[22] 'The matrix format [i.e. the implementation] also represents mathematically the metaphoric rack/card concept previously discussed.'

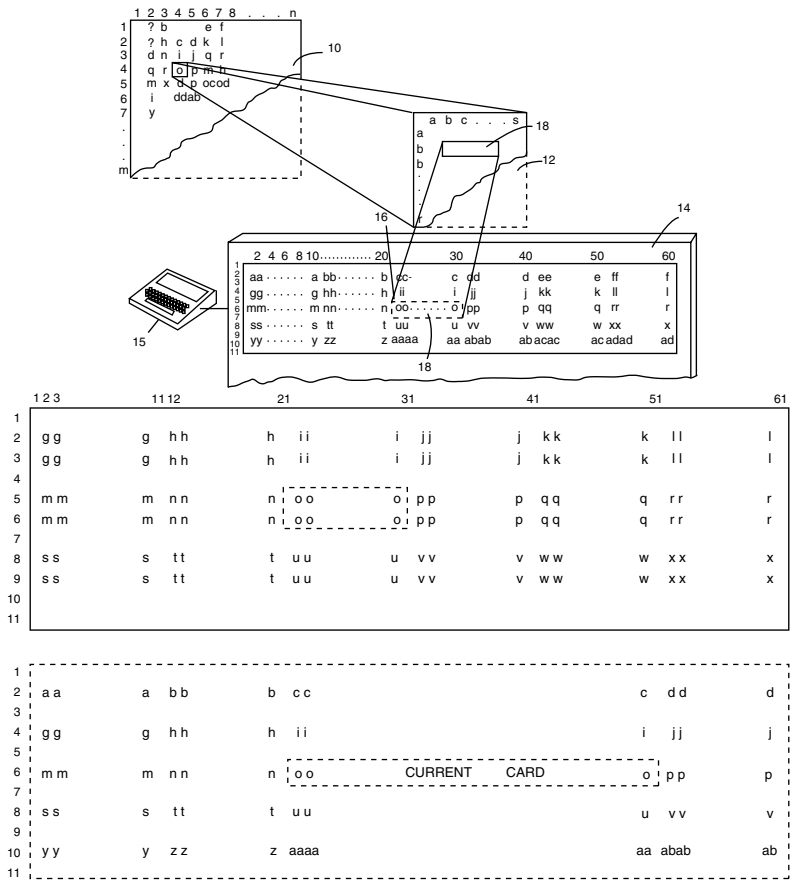

Fig. 4.1. The Zoomracks implementation

The dispute

Those who have followed the Hypercard/Zoomracks descriptions so far may be able to predict the problem in law: did Heckel's patent, which appeared to claim an implementation of a zooming mechanism, actually cover an implementation of cards and links? The League for Programming Freedom thought not and pointed out that Heckel had not seen that his patent could cover the HyperCard model. According to Richard Stallman:

And evidently Paul Heckel couldn't either because when he first saw hypercard it didn't occur to him that it would be covered by his patent. It wasn't similar to his

program, it wasn't similar to his ideas as he understood them but his lawyer told him that the patent could be read as covering things hypercard did. So at that point he started threatening Apple, and when Apple didn't act scared enough then he started threatening Apple's users, Apple's customers, threatening to sue them. And Heckel and Apple made some kind of settlement which is secret, we don't know how much money he squeezed out of Apple this way.

Once when I gave this speech Heckel was in the audience and at this point, true to his name, he jumped up and said: 'That's not true! I just didn't understand the scope of my protection!'[23]

The problem can be seen – in terms of this author's argument – primarily that there is no clear claiming methodology which was available to make Heckel's software claims read explicitly. There was an idea laid out but, because examination was carried out under the regime of looking for the 'machine', it is difficult to extricate and decide upon a scope for the various software ideas which appear in the application. Heckel had a 'virtual' metaphor for what his program did, but the specification did not relate to that metaphor, rather to the implementation: the latter was an attempt to claim a machine rather than the idea which was behind the machine. Thus, the claims effectively allowed coverage of ideas which were not at the core of the invention, since they were 'tied in' to the originating idea.

What is meant by a 'metaphor' anyway? Is it an idea? Is it abstract or – in terms of a programming design – concrete? The concept of 'programming metaphor' is used in the research literature of computer science[24] and a programmer – a person skilled in the art – would understand what is meant, but it is a difficult concept to define.

The current position of lacking accurate software-claiming methodologies is due to the history of the development of software patents, i.e. through the back door rather than reasoned discussion about how best to formulate claiming structures for the new technology. This makes it difficult for the reader (e.g. someone wishing to 'work around' the patent) to extract from these claims just which part of the software 'process' is at the heart of the patent. Thus, what was it that Heckel was really trying to protect in his '308 patent? Was it:

[23] R. Stallman, 'On Software Patents', talk given in Helsinki, available online at http://db.cs.helsinki.fi/~jhuhta/stallman.html.en.

[24] This is particularly used in programming design research. For example, see D. H. H. Ingalls, 'The Smalltalk-76 Programming System Design and Implementation', in *Conference Record of the Fifth Annual ACM Symposium On Principles Of Programming Languages* (Arizona: Tucson, 23–25 January, 1978) available online at http://users.ipa.net/~dwighth/smalltalk/St76/Smalltalk76ProgrammingSystem.html. Smalltalk was a very important early use of the 'object-oriented' metaphor. Once again, this was a Xerox PARC project.

- The metaphor of zooming back and forward into files?
- The card and rack metaphor?
- An implementational method for the card and rack metaphor?
- A specific program (algorithm, perhaps) for implementing one of the above?
- All of the above?

The metaphors certainly appear in the specifications but only as helpful descriptive explanations and are not contained in the claims at all. If they had been claimed, would this have made Heckel's patent stronger over HyperCard? Without claiming it, was the patent readable as much broader than originally conceived? Too broad?

Assuming that Heckel's 1984 application was not defeated by prior art (and there appears to be no suggestion that the Xerox PARC idea was made public before this date) and was inventive (as surely it appears to have been) is it an invention over HyperCard? If Heckel really did not 'understand the scope of his protection', was it that the protection he felt the patent gave was for a different inventive idea? Perhaps if there had been claim mechanisms for software which more clearly delineated what a software inventor was laying claim to, then it might have been possible for Heckel to have more explicitly laid out what it was that he believed was to be protected. These kinds of questions are important when courts have to decide how extensively a patent should be construed and what value is placed upon the description and upon the claims – with, for example, the US doctrine of equivalence and the related European (and common law) notion of purposive construction. Heckel apparently felt that the claims of the original specification did not fully explicate the invention and a reissued patent with amended claims appears in 2000.[25]

Heckel did not disappear from the patent scene after the Zoomracks/HyperCard episode. He was granted a HyperCard-related patent in 1993[26] and also appeared in 2003 in an attempt to sue newspapers which he felt were infringing his reissued '308 patent.[27] His company, previously

[25] Reissue 36,653. A co-inventor was added to the re-issued patent.

[26] US 5,228,123, 'Interface and Application Development Management System Based on a Gene Metaphor'.

[27] 'A Lee Enterprises subsidiary is fighting a California inventor who earlier this year filed suits against newspapers he contends are violating his patents governing how Web sites are designed. Lee unit TownNews.com, a Web hosting and design firm, is defending itself against infringement suits Los Altos inventor Paul C. Heckel filed against two newspapers it hosts ... Heckel claims newspapers that display abstracts of full stories that comprise multiple print columns violate his patents. The technique is a popular method to tease Web site visitors to read further.' *Newspapers and Technology* (May 2003). The *Editor and Publisher* (17 June 2003) later stated that: 'A Los Altos, Calif., inventor has withdrawn a patent infringement lawsuit against a dozen small newspapers, but said

a software developer, has appeared to move from software development to IPR licensing.[28]

Prior art and persuasion: legal document drafting

The role of the patent attorney – and this is *the* primary role for which he receives payment – is to get as much protection for his client as possible and, as mentioned in the quotation at the start of this chapter, this involves an interaction between representative and examiner on what the technology is and how this relates to the matter in the current application's claims. This is mediated in a context which is set by the search report and, therefore, the prior art which is discovered in search is vital to ensure good examination. Without good prior art the whole process of examination becomes liable to the examiner making decisions on poor technical grounds, and the good patent attorney will always be looking for weaknesses in examination which can be turned to the benefit of the client. In an earlier study – prior to the relaxing of the examination of computer-implemented inventions – it was reported by one patent attorney with a German computer firm that it was sometimes possible (particularly when examiners were outwith the main computer technical field) to confuse them:

> If you are talking to examiners who specialise in the field of GO6F ... the patent examiners there are very much aware of the current case law from the EPO Boards of Appeal and it is very difficult to pull wool over their eyes. With newer ones, or less cynical ones you can. I mean, older examiners also tend to be much more restrictive than younger examiners. With older examiners you certainly can't pull the wool over their eyes, sometimes with the younger ones you can. When you are talking about examiners who specialise in other technical fields, in which computers are increasingly being used they are certainly not as aware of the case law of the EPO Boards of Appeal. In many cases, as a patent attorney, you are obliged to cite the relevant case law at them because their first attitude is to look in the negative catalogue of Article 54 and to say, computer programs are not patentable, I'm going to reject your invention. I then come back with [a list of cases] ... Then they often turn completely to the other direction and can have the wool pulled over their eyes without really thinking is there a technical problem behind this, is the attorney just clothing the claim in useless hardware features in order to disguise the software invention (the non-patentable software invention behind it). Those examiners are much more susceptible to manipulation.[29]

newspapers haven't heard the last of him. Paul C. Heckel said he plans to refile claims against papers that allegedly violate his inventions once his company, Quickview Systems, which owns the patent, returns to good standing.'

[28] 'Quickview Systems – A Technology Licensing Corporation' – see at http://www.quickviewsystems.com.

[29] Leith, *Harmonisation of Intellectual Property in Europe*, p. 82.

Since this interview took place, a considerable amount has happened with respect to examination of software and we would expect that the basic problems of understanding the case law would, for many examiners, now have gone. However, this attorney's point was also that there was a problem when examiners were carrying out an examination which was outside their specialist areas. Clearly, some applications will arrive at the EPO which are in fields which are rarely encountered, or which transcend the borders of several fields and will cause possible examination headaches for the individual who has to search and also the individual who has to examine.[30] The example here of legal document assembly appears to be one such technical field.

Method and system for creating documents

The example relates to a technology which lawyers should also understand and raises different issues from those raised by the Heckel patent, primarily issues of searching through prior art during examination. This example is European patent (EP0988609), which was filed by Ian Woodcock in June 1998 and granted in September 2002, entitled 'Method of and system for creating documents'. The field of the invention was the preparation of documents by computer were related to a method (claim 1) and a system (claim 2).

Lawyers will know well that document creation is time consuming, and that ways to speed this process up – particularly for documents which are relatively 'standard' – can be cost effective in the modern legal practice. Woodcock suggested that his invention could be applied to documents such as wills, powers of attorney, self-assessment tax, contracts of employment, employee records, etc., indicating that the method has potentially wide-ranging applications. Producing documents by computer was well known: the word processor had, by the 1980s, already transformed legal practice by removing the need to type out and carefully check multiple copies of documents. What Woodcock claimed for his invention was the automation by means of interactive questions, so that a list of questions would be produced which depended upon the answer to the prior questions. The final document could be constructed from pre-prepared clauses, these being chosen to reflect the answers (much like picking these from the traditional text containing legal precedents). The patent contains several flowcharts and programming information

[30] Examination is by a team, but in practice to be left to one member of the team to be primarily responsible for each application – Art. 18(2) EPC. Supervision of examiners is also present.

detailing generally well-known techniques for carrying out this kind of processing.

The original search report completed for the patent application[31] noted that there were two relevant priority documents which were classified as category 'X' for all the relevant claims – that is, that the documents were 'of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone'. These were US 5,272,623, entitled 'Software programming method for forming Government contracting documents' (granted December 1993) and EP376695, entitled 'Operator-defined dialog boxes and processing logic for computer systems' (the latter was an application later deemed withdrawn). Neither of these two documents were in the field of legal document assembly but both involved the construction of questionnaires.

A search report which produced such negative results indicating that all of the claimed aspects of the invention were to be found in the prior art would be decisive in refusing grant, so the examiner was obliged to request 'observations' from the patentee within four months.[32] Woodcock's representative filed these observations and an amended version of the patent, reducing this to two claims which acknowledged the known features to be found in the two cited pieces of prior art. Example screen shots were also forwarded but these do not appear in the on-line filing system due to poor quality. The arguments made were that:

1. The new claims had been amended by the use of the term 'interpretative' to quality the expression 'logical argument and branches' (i.e. that artificial intelligence techniques were being used) and the term 'continuous' to qualify the questionnaire.
2. The difference between US 5,272,623 and the current application was that that prior art patent differed since with word processing macros there is no real provision for moving backwards, since this requires a user to remember and retrace all the questions and answers, so, in practice, the data has to be re-entered.

The examiner replied that the proposed amendments introduced new subject matter since 'the artificial intelligence activities of the questions database described ... do not suffice to rend the definition of "the interpretative logical argument and branches" as introduced in the claims, whose wording appear also vague and unclear'. The applicant's representatives replied (in August 2001), outlining that 'logical argument' as a term was 'well understood', that this was fully supported in the original

[31] October 1998. [32] EPC 96(2) and Rule 51(2).

text and that there were other distinct differences between the prior art cited and this application.

We see here the persuasive nature of the patent attorney. Given a very poor start with a near-damning search report, the attorney was still able to produce a revised application and arguments as to why this revision was worthy of patent protection. The persuasion was sucessful and the examiner noted in November 2001 that he intended to grant a patent on the basis of the revised documentation. The attorney had not achieved all that was requested at the beginning of the application process, but no doubt the client was pleased that, out of very little, a successful patent had been constructed.

Was it novel and inventive?

This author assumes that this was not a novel idea and also lacked inventive step – a view taken as someone who might be described as a person 'skilled in the art'.[33] This may well be incorrect and while there may indeed be a patentable invention at the core of this specification, there are certainly reasons to consider that there were problems with the examination. For example, the search report covered only patent documentation and did not reflect the research and commercial developments which had taken place in the 1970s to 1990s in the field of legal document assembly. Given this limited search report and a non-critical examiner who appeared to be easily swayed by pseudo-technical arguments, we might suggest there was wool and there were eyes involved in the examination process.

The field of legal document assembly was well known by 1998. Indeed, some of the original developments in the field were carried out by Jim Sprowl (ironically a well-known expert in patent and computer law[34]), who carried out work for the American Bar Association in the 1970s. His article 'The Automated Assembly of Legal Documents'[35] clearly describes the processing language *ABF*, a system which is based upon just the 'logical' processing outlined in Woodcock's patent. Of this system Sprowl wrote:

[33] See Leith and Hoey, *The Computerised Lawyer*. For a number of years in the 1990s and early 2000s, students on the author's LLM in Computer and Law were set assessed programming tasks to do just this kind of document assembly using a variety of programming tools – SQL, PHP, HMTL, etc.

[34] See e.g. P. Maggs, J.T. Soma and J.A. Sprowl, *Computer Law: Cases, Comments, Questions* (Minnesota: West Publishing, 1992).

[35] In B., Niblett, (ed.), *Computer Science and Law: An Advanced Course* (Cambridge: Cambridge University Press, 1980). The full report was published in *American Bar Foundation Research Journal* 1 (1978).

We undertook a second project to design a computational processor that could write its own program using form documents, statutes, and regulations as a guide. No such processor had ever before been constructed, to our knowledge, and at the outset of the investigation it was not at all clear which form such a processor would take if it even proved possible to construct one.

Sprowl's program was able to hold input information and insert it into locations in the document as required (i.e. the input information was held in a short-term 'database') and using a logical structure to request only information from the user which was required (i.e. a dynamic listing of questions) through 'optional and alternative passages':

Such optional and alternative passages may also contain additional optional and alternative passages, which, in turn, may themselves contain additional optional and alternative passages and so on without limitation.

Of course, the program had no GUI interface, given that it was written in the 1970s and ran on a CDC mainframe, so the implementation described by Woodcock where the continuous formatted screen display is an important factor of the invention could not have been implemented in that hardware. Nonetheless, a continuous display of only relevant questions would be presented on the terminal in front of the user. Since Sprowl's early work, a whole host of commercial systems had been produced and were running and being sold by 1998, which basically operated with the same underlying technology as ABF. Lauritsen, who had been involved with one of the early research projects between Harvard and Brigham Young Universities, and Soundakoff outlined a number of these in an article in 1998.[36] Further, in the 1980s a number of projects had been undertaken in the field of 'expert systems' which carried out this task of questioning the user and, based upon the replies given, choosing various paths for other questions. Woodcock notes this, but gives no indication of where his invention differs from this non-cited prior art, indeed seeming to use the fact that the few pieces of prior art cited in the search report are proof of his overall novelty.[37]

[36] M. Lauritsen and A. Soudakoff, 'Power Tools for Document Preparation' (1998), at http://www.capstonepractice.com/amlaw6.pdf. A version of this had earlier appeared in *AmLaw Tech, The American Lawyer's Technology Magazine*.

[37] An article by Jim Sprowl had been cited as background material (i.e. 'A') in the search report for EP0376695 which was cited in the search report for Woodcock's patent but was not mentioned by the examiner. The file documentation for EP0376695 has been destroyed in line with the workflow practices under Rule 95a which requires files relating to refused, withdrawn or revoked applications to be destroyed after five years. Such destruction does not appear to be necessary – Rule 95a(3) states that electronic versions will be considered to be originals – given that digital files effectively require minimal cost to store and may be of use to researchers (both academic and patent searchers) in future years.

There was no backward movement through the questions in ABF, a point claimed by Woodcock's attorney as inventive over the prior art in the letter of January 2001, but the patent in any event lays no claim nor describes a mechanism for moving backwards and changing values input when, for example, a user inputs information in error and wishes to change that input. In some ways, this latter element might be seen to be inventive but it also, to a programmer, does appear to be a relatively trivial problem. It is a standard programming problem rather than one requiring great inventive leaps once one has a mechanism for storing and inserting data into locations as required (as had ABF), and indeed many of the commercial systems outlined by Lauritsen were linked to databases of various sorts. Any programmer who was given a specification for an interface which required 'back stepping' could implement it in a number of ways – storing input on a tree structure and pruning branches as required would be one simple means. If Woodstock had actually claimed 'back stepping' as a method in his patent, would he have effectively monopolised all possible programming solutions or only the method by which he demonstrated implementation?

Of course, when we move away from the specific technical field of legal document assembly, the prior art is full of such back-stepping procedures. The most well known of all must be the 'undo' and 'redo' facility in Microsoft Word.[38]

What – if anything – went wrong?

Once again, presuming that this patent was incorrectly granted – probably an assumption with which the patentee would disagree – what might have been the problem? The search examiner produced clear evidence that there was prior art within the patent literature which was felt to cover the application. The only materials cited in the search report were patents – there was no use of journal or other written materials, and no mention of any of the commercial programs which were on offer. It is as though a whole body of prior art was missing. Did the search examiner simply believe that he had found enough to deny grant and that there was little

[38] This was pointed out to the author by a respondent on the Law-AI mailing list, proving that the many obvious exemplars are often overlooked. See also the application filed December 2004, EP1491997: Undo infrastructure, 'Methods, systems, and computer program products that automatically generate and track undo information so that the developer of a user interface object need not be responsible for generating and tracking undo information. Change notifications for changes to an object within a visual user interface designer are processed . . .'

need to do any further time-consuming research into databases or materials to which he did not have easy access?[39]

It is not possible from the European Patent Office documentation to discover what the main technical field of the primary examiner is. It is possible to determine a name from the filing documentation, but unlike US patents there is no examiner name cited on the actual patent which means that we cannot easily discover the examiner's main expertise through a quick search of patent documentation. It is possible, perhaps, to suggest that it was not in the field of legal document assembly.

The finding of inventive step (the 'problem and solution' approach) is founded on the notion that the relevant prior art has been determined. In a situation such as software where the prior art may have been a program written in the 1970s or 1980s for which the documentation has been lost or resides in a cupboard somewhere, or where the documentation is simply the source code of the program, how is the examiner to be enabled to determine that prior art? Clearly, in this example there is a substantial amount of prior art which the man 'skilled in the art' would be aware of, yet which appears to be hidden from the view of the examiner. If the relevant prior art is not found, then an invention can be incorrectly assumed over the found prior art.

The actual examination situation is worse than this, because there is a presumption that only prior art which is available is actually prior art (see, for example, T0144/95[40]) and there is much in the way of program source code and object code which exists but not in any usable format by the examiner – that is, it is most unlikely to be indexed, well documented, or available for easy inspection.[41] Given, too, that most code has not been circulated, the actual quantity of textual material in the public domain will

[39] See discussion of the procedural split between examination and search and its recombination under the Best programme in Leith, *Harmonisation of Intellectual Property in Europe*. This has now been revised and DG1 and DG2 have been combined into DG1 where both search and examination take place.

[40] 'It is true that the case law of the boards of appeal clearly indicates that the theoretical possibility of access to information renders it available to the public. This is certainly true of a document in a library or a specific device in a place accessible to the public. This principle however can only be applied after the document or the device has been identified so that its information can be unequivocally defined. If such an identification is impossible, i.e. if it is uncertain that the document or device ever existed or it is uncertain in which precise form it existed, then the document or device cannot be used for patent purposes.' T0144/95, 26 February 1999.

[41] J. Weyand and H. Haase, 'Patenting Computer Programs: new challenges', *IIC* 36 (2005), 647–63 suggest that obligatory lodging of source code would 'provide a reliable basis to ensure that [CCI] . . . are not patented without disclosure and transparency'. This would provide prior art from only those inventions for which protection had been requested but would also impinge upon other trade secret rights or the reverse engineering prohibition. It neither seems practical nor covers the parallel conceptual nature of software as outlined here in Chapter 2 (e.g. protection of the metaphor).

be very much less than that which is represented by 'know-how' (which comprises part of the skill in the art). Even that code which has been made public may well be in such an obscure location that it cannot be found – the author's own Ph.D. thesis describes an InterLisp program from the early 1980s (it is not, to my knowledge, currently possible to access a system which will execute this program) and is thus 'public' but the only versions of the code lie in my office in a now-aging listing and on a magnetic tape. How is the examiner to include such know-how in the process? Only if they have sufficient relevant expertise in programming techniques, perhaps?

Macrossan and more document assembly: levels of technical contribution

Patents such as that of Woodcock lie close to the border between software and 'business method' patents, but the specification of the patent does not reflect that there exists prior art in the business field which relates to the manual construction of legal documents through the use of precedent books.[42] A criticism of many similar patents is that there is actually no novelty at all – that they simply take well-known methods and claim novelty by adding computerisation. (A similar criticism was made of patent offices which granted protection on devices utilising transistors when the technology moved away from that based on valves.) It appears that some offices are stricter in denying the mechanisation of well-known tasks than others (the UK office appearing to be more strict than the EPO). But while one might have thought that the EPO examiner in this patent would have raised these issues, the filed documentation does not show it having been seen as a potentially important issue at all. One wonders, therefore, whether the EPO is moving closer to the US Office, which latter appears happy to provide protection for well-known methods newly computerised.[43]

Macrossan's application (WO0242953[44]) concerned a method of constructing 'corporate entities' over the internet and is particularly interesting,

[42] Many legal publishers provide a number of texts containing standard precedent clauses which can be manually put together into a single document such as a contract or licence.

[43] For an example of this which relates to patent documents, we can see US 7,076,439 which relates a method of 'managing projects' – the implementation concerning a system which keeps details of patent applications, deadlines and then sends reminders to the individuals involved providing 'options not available on the conventional docketing systems, such as automatically increasing the frequency with which reminders are sent as the deadline approaches, and automatically increasing the number of individuals to whom the reminders are sent as the deadlines draw near'. This appears to be a very well-known technique and thus surely hardly novel.

[44] The PCT request for examination was made at both the EPO and the GB offices, working from the same application document. An Australian patent (AU759130B) has been granted.

given that it was subject to consideration in both the London High Court and Court of Appeal (which we look at in Chapter 5). The general invention follows the schema of the document creation application above: questions are answered by a user at a remote browser, saved in a database and then – when sufficient information has been collected – the legal documents to form the entity are printed out or electronically submitted. The application was made in November 2001 from an earlier filing of November 2000, a period where the internet was important in terms of business opportunity. The international search report cited four pieces of prior art which the search examiner considered to be documents 'of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is taken alone'. The technology outlined in the application is of little novelty – for example, the list of errors which are checked for include 'entry ... of a negative amount in a field which may only validly accept a positive amount' is hardly – given the history of data processing at that point in time – a major advancement. However, the application also lays out more advanced error checking where the statements input by the user are used to check, for example, whether the user has input sufficient resident directors legally required by the local regulations.

The application contains some fifty pages of program description, laying out clearly the functionality and the operating methodology of the program, and also contains sixteen pages of diagrams, which include hardware, screen shots and flowcharts. Given the detail in the application, few could argue that Art. 83 EPC (Disclosure of the Invention) – 'must disclose the invention in a manner sufficiently clear and complete for it to be carried out be a person skilled in the art' – has not been more than sufficiently met. The system at the heart of the application is in commercial operation (UKCorporator.com) and one of its co-owners, when asked why the patent was being sought, answered:

> We haven't turned our minds much to what should or should not be the state of the law in this area. Rather we have focused on the current state of the case law, and the relevant statute, and concluded that our subject matter is patentable. We also noticed these three granted UK/EU patents and concluded that if they are patentable then our subject matter is even more so –
> - GB2373624 (a UK patent, granted in or after 2002, for an Automated Online System for Generating Exam Questions);
> - GB2345997 (a UK patent, granted in or after 2002, for an Automated Online Dispute Resolution System);
> - WO9506294 (a EU patent, granted in the mid 1990s, for an Automated Online Loan Application System).[45]

[45] Online at http://www.emailbattles.com, dated 9 June 2006.

The application was filed through the international PCT route and examination was requested at both the UK Office and the EPO, the latter – at the time of writing – appearing to raise the same issues which were raised in the UK office. The hearing officer in the UK had rejected the patent application,[46] relying on all three of the commonly raised exclusions in the context of software-related business patents:

- the mental act exclusion;
- the computer program exclusion;
- the business method exclusion.

Some discussion of novelty and obviousness was also raised by the examiner, but only the questions of whether the application could proceed to those issues were at the core of the hearing officer's decision.[47] The hearing officer noted that the three patents (above) were brought to her attention, but that these were irrelevant, since each case had to be decided upon its own facts. She found that all three exclusions were appropriate, and then moved on to deciding whether they were relevant 'as such' – that is, was there a 'technical contribution' which removed them from the excluded categories? The applicant's arguments were various, but essentially can be reduced to one which suggests that a program carrying out a task is necessarily technical.[48] The applicant's arguments did not persuade the hearing officer, who concluded:

> I have tried my utmost but I simply cannot see that the method is other than a computer system programmed in such a way as to provide a guided question session in response to answers given by a user and to then provide expert advice to the user in the sense of generating documents in an electronic format.

and also put the application into the 'business method' basket:

> Whilst there may be economic benefits and reduced risks in computerizing the production of these legal documents, I do not consider that problem to be a

[46] BL O/078/05 March 2005. GB application, GB 0314464.9

[47] Little wonder that novelty and obviousness seemed pertinent. Claim 1 is: '1. A method for assisting in the formation of a corporate entity, in an answering session, said method comprising the steps of : (a) interactively communicating via an interactive communications device, a series of questions to a user attempting to establish said corporate entity; (b) permitting said user to provide an answer to said set of one or more questions; (c) successively selecting a further set of one or more questions for display to said user, at least one of said selection being dependent upon said user's previous answer or combination of previous answers; (d) permitting said user to answer said successive set of one or more selected questions; (e) repeating steps (c) to (d) until said user has provided enough information to allow for the determination as to which documents are legally required for the formation of the said corporate entity; and (f) generating the legally required documents referred to in step (e) in an electronic form.'

[48] The applicant also suggested that a 'pre-saving algorithm' was a technical contribution, rather than saving after the information was input.

technical one. It is a problem which I consider to be a business or administrative problem and it is solved by providing an automated tool to stand in for a legal expert. Whilst these outcomes may all be highly desirable, desirability is not the test an invention must pass to avoid the exclusions. However desirable it may be, I can see no technical contribution provided by the invention.

If Macrossan's application had been part of a system which gathered information and then produced a legally correct method to control a manufacturing process (soup, perhaps[49]) or an aircraft, then would it have been excluded from protection? It seems unlikely that it would have been refused by the patent office if it had been part of such a mechanistic operation. We do see, though, that applications of a similar kind to that of Macrossan are granted. For example, the Menashe patent which was discussed in Chapter 1 is as tenuously 'mechanical' as that of Macrossan and perhaps explains the attitude of the applicant which seems to be one of 'if they got protection, I should too'. Indeed, one of Macrossan's complaints (when the refusal was heard at appeal) was that there was a bias against his application from the UK Patent Office based upon an alleged secret submission made to the hearing officer by the examiner and also based upon a Patent Office publication referring to the Patent Office's '... strong tradition of rejecting patent applications for software ... this tough approach has ensured that only patents with a 'high presumption of validity' are granted', the argument being that the application was being dealt with more harshly than other applications.

The primary discussion in procedural matters is usually that different offices have different levels of inventive step – a fact which is well accepted. It may be, though, that there are other differing levels – for example, the level of what a 'technical contribution' is, rather than this being a simple black/white dividing line. The Hearing Office from the UK Patent Office appeared to be applying a relatively high level of requirement as to 'technical contribution' in support of the examiner's view, yet other examiners must be applying a lower level or applications such as Menashe would be difficult to prosecute through to grant.

Is this a surprise? Not really, if we consider the quotation at the start of this chapter: pushing the merits of an invention is a matter of psychology rather than technology. This must imply that patent attorneys, in pursuit of protection for their clients, will – as part of their tactical toolkit – focus on those offices which provide the lowest level with regard to 'technical contribution' – which is unlikely at the present time to include the UK office, this (anecdotally) appearing to have a higher level of technical

[49] As suggested in *Patent Applications by CFPH LLC* [2005] EWHC 1589 (Pat) and discussed below.

contribution than the EPO.[50] They would also be expected to highlight the differences in approach between national offices, courts and the EPO as they apply pressure to push developments in a more 'client friendly' direction.

We return to Macrossan in more detail in Chapter 5, but for now point out that we have an 'invention' which – if it is novel and inventive over the prior art – is being excluded from protection for reasons which are not related to it as a technological artefact (which it surely is) but because this is a task at its core which can be carried out by a human as a mental method. There appears to be very little difference between this application and that of Menashe's gaming system: both systems could be represented by the type of diagram used by Menashe (see. Fig. 1.1) and both offer similar types of advantages to their users, one in gaming (surely a mental method?) and one in document preparation. The question is: how does an examiner decide when it is correct to apply this exclusion, and how do we know it is being carried out consistently across the EPC jurisdiction?

Evidential matters: do the patented ideas actually work?

It is a fundamental assumption of patent law that ideas which are protected should be workable – thus perpetual motion machines are not protectable, since they go against well-developed physical theory.[51] Evidence can be taken to prove the workability of ideas and Henry Ford's litigation over the Selden patents, it has been suggested, was only terminated when the claimed invention was required by the court to be built during litigation[52] and proved to be ineffective. The rules of the EPC allow patent examiners to request evidence to be presented but the guidelines for examination note that evidence is usually taken only in opposition,[53] which suggests that the examiner usually accepts the word of the applicant that the claimed invention is workable. The examiner will

[50] There is no need to use the UK Office at all in order to get protection in the UK, since a successful European examination will provide a European UK patent.

[51] 'One further class of "invention" which would be excluded, however, would be articles or processes alleged to operate in a manner clearly contrary to well-established physical laws, e.g. a perpetual motion machine.' *Guidelines for Examination, PART C, IV, 4.1 General remarks*. Also see T 5/86 '*NEWMAN/Perpetual motion*' [1988] EPOR 301. The reason for refusal was 'insufficiency of description'.

[52] W. Greenleaf, *Monopoly on Wheels: Henry Ford and the Selden Automobile Patent* (Detroit: Wayne State University Press, 1961).

[53] 'Formal taking of evidence in accordance with Rule 72(1) will occur mainly in opposition proceedings and hardly ever before the Examining Division. The following Sections of this Chapter are therefore based primarily on opposition proceedings. However, they also apply *mutatis mutandis* to other proceedings and particularly to substantive examination.' Part E, IV, 1.

be relatively expert in the field in which he is operating and, in many fields (chemistry excepted, due to the difficulty of predicting behaviour from structure), this expertise allows the examiner to 'have a feel' for the believability of the application and see those which, for example, operate against physical laws. In the examination of software, this is perhaps an assumption which is more difficult to make than in the more traditional fields. Why should this be? Primarily because – as outlined in Chapter 2 and expressed by Moor – software is describable as various artefacts: theory, model and/or code. Code is fine, model can be more problematic, but theory is dangerous territory for examination.

Unfortunately for the examination process, much of what is published in computer science literature, and particularly in that emanating from the artificial intelligence community, is description of a theory of how something should operate. This can often be an academic article which outlines a project which is to be followed[54] or, indeed, there can be dispute over whether what is being described is actually to be found in the program from which this was coded. A leading example from the literature of this was the debate over whether a program (AM) was capable of discovering mathematical proofs.[55] But other areas of computing, too, have had critics[56] who have suggested that the programs described do not work in the manner in which they have been claimed to work. The culture of computing differs from that of science and engineering: in the latter there is pre-disposition to require evidence to be submitted to the relevant community of claimed advances in technology. In computing, there often seems to be a sense that theories can be easily transferred into code – that is, that though it claims to be a science, it does not necessarily follow the classical scientific method. For example, very few research programs are ever made available to others for examination and independent testing and verification.

The computer is not only attractive to AI researchers, but also to those with even more esoteric views on life and it will be expected that the computing equivalent of 'perpetual motion' applications will be received at the various offices, and some will be granted.[57]

[54] Usually including, 'This will . . .' rather than 'This did . . .'.

[55] G. Ritchie and F. Hanna, 'AM: A case study in methodology', *Artificial Intelligence* 23 (1984).

[56] Including this author. See P. Leith, *Formalism in AI and Computer Science* (London: Ellis Horwood/Simon and Schuster, 1990).

[57] See US 5,734,795, granted March 1998. '1. A computer-based system for allowing a person to experience systems of mythology within virtual reality environments generated by the computer system, comprising: a) an existential analyzer module (EAM), for assessing a portion of a person's meaningful experiential world, ...'

In part this problem arises because the malleable nature of computing appears to offer much to those from other fields: for example, the cognitive psychology community has viewed programs as methods of proving psychological theories such as that reasoning is rule-based[58] and we have seen applications from the artificial intelligence field which use models based on rules or brain organisation.[59] No doubt, their advocates insist that these are not theories, but the evidence of the past decades has been that artificial intelligence artefacts have not taken over the marketplace.

A more mainstream example of recent work being carried where there is debate over the success of the theoretical model is Latent Semantic Indexing, a technique which has been around for some decades and involves giving mathematical descriptions to documents depending upon their contents. Critics have suggested that the method is still not particularly effective, though at least one commercial firm is advertising patent search using the technique.[60]

Is this any different from the other more traditional fields? Perhaps not, after all, Selden's patent was awarded for an invention that was – when built – effectively a failure as a motorised vehicle – but in the examination of more traditional fields it is possible for an expert examiner to intuitively sense the weakness of the application. In software, the breadth of areas where inventions are being claimed and the lack of a large body of prior art (to provide evidence as to what is possible) may lead to examination by those who are more receptive to the idea than they should be.

This should not be taken to imply that the consideration of evidence is a simple task or that a greater quantity of evidence will solve any examination problems. In earlier research, it was suggested that presenting evidence to examiners can sometimes be a very successful way of confusing them. One German patent attorney stated:

> The technicians who usually run the oppositions departments are not used to witnesses and they want to avoid it if they can. It is really a little bit of a trick that you bring in – as an opponent – something which can only be proved by witnesses (i.e. public use) in order to force the opposition division to look in more detail to your written prior art arguments . . . so that they don't have to hear the witnesses.[61]

[58] The early work of Newell and Simon gave huge impetus to the expert system model.

[59] Neural networks have been popular areas for patent applications, yet the field too is considered problematic by sceptics.

[60] 'PatentCafe's Latent Semantic Analysis search engine is recognized as the industry's most advanced concept search technology.' See http://www.patentcafe.com. In the 1980s and 1990s many commercial companies sold software as 'expert systems' or including artificial intelligence techniques but that is hardly proof of technical success: more possibly it is proof of imaginative marketing.

[61] Leith, *Harmonisation of Intellectual Property in Europe* p. 96.

Such a tactic would appear to be particularly relevant and useful in software where 'public use' is perhaps more likely to be found than a well-documented description of the invention.

Classification system developments

The importance of a good classification system can be seen by WIPO's description of its International Patent Classification system (the IPC):

> 6. The Classification, being a means for obtaining an internationally uniform classification of patent documents, has as its primary purpose the establishment of an effective search tool for the retrieval of patent documents by intellectual property offices and other users, in order to establish the novelty and evaluate the inventive step or non-obviousness (including the assessment of technical advance and useful results or utility) of technical disclosures in patent applications.
> 7. The Classification, furthermore, has the important purposes of serving as:
> (a) an instrument for the orderly arrangement of patent documents in order to facilitate access to the technological and legal information contained therein;
> (b) a basis for selective dissemination of information to all users of patent information;
> (c) a basis for investigating the state of the art in given fields of technology;
> (d) a basis for the preparation of industrial property statistics which in turn permit the assessment of technological development in various areas.[62]

The classification system allows examiners to apply one or more codings to applications as they arrive which describe the contents of the application.[63] Unfortunately, from the point of view of software examination, the European classification systems[64] are primarily located in the hardware view of invention and do not reflect what we have been describing as the 'programmer's view' – a programmer involved in 'operating systems' or 'database design' would not find classifications which were explicitly appropriate for their fields: instead, they would have to look throughout several classifications (G06F would have been the primary classification, however) to find relevant prior art. The lack of such a coherent description of applications and specifications explains why it has been so difficult to produce statistics on software patenting and in part explains one of the potential difficulties in software examination – that of finding relevant prior art within the patent corpus.

[62] WIPO, *International Patent Classification Guide*, 8th edn (2006).
[63] This is clearly a time-consuming but important task and there have been attempts to automate this. See M. Krier and F. Zaccà, 'Automatic categorization applications at the European patent office', *World Patent Information* 24 (2002), 187–96.
[64] A combination of the IPC and a more detailed European ECLA.

Indeed, the revision to the IPC in 2006 shows just such a development in classification techniques, with a slightly more software-oriented approach to invention, but a more appropriate classification might enable even better access. For example, the robot control program which was described in Chapter 2, we might consider to be classed somewhere in G06F relating to 'programming languages', but, rather than being viewed as software, such systems are currently set in other fields – such as B25J (manipulators) and G05B (control or regulating systems).[65] There is, in fact, no classification which covers the computer scientist's 'programming language design',[66] the closest being G06F 9/44.[67] There are approximately 800 granted EP patents in this latter sub-field, with a relatively wide spread of type which – it is suggested – would not appear to a computer scientist to be a particularly 'logical' taxonomy. Is this a major difficulty? Probably not in examination, since an experienced examiner will have an understanding of where relevant prior art will be found. However, it may be problematic in terms of enabling competitors to determine whether an area is free to work by doing a search of patent documentation – particularly if SMEs wishing to avoid infringement are involved.

A first classification may not be the most appropriate and this can be amended through prosecution of the application. However, if patent applications are to lie unexamined for several years after publication (as is the current practice due to workloads) then prior art searchers will find these difficult to access and thus to work around.

The US office has a 'Class 705' to reflect the new information industry inventions (i.e. 'business methods')[68] and the revised 2006 IPC classification system now has a similar class (G06Q), which does give us more useful insights into patenting activity from applicants and demonstrates that, as of

[65] See, e.g., EP1700175, 'Device And Method For Programming An Industrial Robot,' published September 2006. The fields searched were G05B, B25 J and G02B. (Abstract: 'The invention relates to a method for programming an industrial robot by means of a simulation program. According to said method, control commands are entered using a manual programming device and said commands are displayed in visual form on a screen as displacement and/or processing operations of the robot, based on the data of the latter. An object to be manipulated is likewise represented on the screen and a three-dimensional image of the robot and the object is reproduced.')

[66] For the terminology as used by those 'skilled in the art' see ACM Sigplan special interest group at http://acm.org/sigplan/.

[67] Also related is 9/44 'Compilation or interpretation of high level programme languages'. See for example EP140184 (A1) 'Programming Language Extensions for Processing Xml Objects And Related Applications', filed June 2002. There are just over 100 EP patents with this classification (as of late 2006).

[68] Class 705 Data Processing: Financial, Business Practice, Management, Or Cost/Price Determination. See www.uspto.gov/web/offices/ac/ido/oeip/taf/def/705.htm

November 2006, there are almost 2,000 applications in field G06Q – '*Data Processing Systems Or Methods, Specially Adapted For Administrative, Commercial, Financial, Managerial, Supervisory Or Forecasting Purposes; Systems Or Methods Specially Adapted For Administrative, Commercial, Financial, Managerial, Supervisory Or Forecasting Purposes, Not Otherwise Provided For*' – which mirrors the US classification 705.

Problems and solutions

At the heart of the EPO's examination of applications for obviousness lies the problem and solution approach. This is a formalised attempt by the EPO to provide a mechanism to make more objective the determination of inventive step. Without such a formalistic approach it would be difficult to harmonise both across and inside the various technical fields:

> In the problem-and-solution approach, there are three main stages:
> (i) determining the 'closest prior art',
> (ii) establishing the 'objective technical problem' to be solved, and
> (iii) considering whether or not the claimed invention, starting from the closest prior art and the objective technical problem, would have been obvious to the skilled person.[69]

There are two main problems which might arise when using this approach with computer-related inventions. First, if the classification does not properly point to documentation in the field, the examiner will have to look across all technical fields rather than a specific sub-field relating to computer science, and he may miss the prior art which is well known to those skilled in the art. Thus, in the example above of the Woodcock patent, we can see that the classification used would not have pointed the examiner towards relevant prior art. It was not only the classification which was applied to the application that was the problem here, perhaps: the main problem may be to do with the classification system as a whole, as discussed in the section above. It is difficult to know the impact of a classification system, since this is all connected with the way that an examiner reasons about an application, and a good examiner must be better able than a less experienced examiner to overcome the types of problems arising from a poor classification system.

Second, in those areas where the prior art documentation may be scarce (for one or more reasons), it is possible that the applicant is given more protection than in areas where there is more prior art documentation. In a

[69] Para 9.8 Part C IV-24 Chapter IV, *Guidelines for Examination in the European Patent Office*.

well-worked technical field we would expect this: an invention where there are lots of other inventions must have lesser scope than where the invention is new and not simply incremental. However, if the prior art is not available to the examiner, it is possible that a larger 'objective technical problem' may be erroneously constructed. For example, in business method applications where there appears to be a dearth of prior art within the EPO search files, it might be possible for an examiner to conclude[70] that the problem is 'how to automate legal document assembly' rather than something more limited, such as 'providing a mechanism for combining document elements'. This, together with the lack of prior art, may give the first solution to this problem very wide scope of protection. Anyone coming afterwards with inventive improvements to this problem will have to negotiate with the owner of the over-broadly scoped patent, which is clearly not desirable.

The problem and solution approach has been used for a number of years but we should expect that for each new technical field there will be a need to apply the approach in a way that takes account of this new technology.

Public input to examination

There has been considerable interest in developing some form of public input to the examination process. In Europe, this has been relatively well developed with the opposition system after grant. In the US there have been calls by many critics of the current system to include such a process, and organisations such as the Public Patent Foundation have been initiated to oppose what they deem to be invalid patents.[71] The primary aim of these calls are to increase the likelihood of prior art being found and then to require re-examination by the Patent Office rather than litigation. In Europe, once the opposition period closes, there is no means for re-examination by the EPO, though there may conceivably be such

[70] It is the role of the examiner to properly determine what is the objective technical problem, not the applicant.

[71] http://www.pubpat.org. As an example of this work we can see their reporting of the JPEG re-examination: 'PUBPAT filed a formal request with the United States Patent and Trademark Office in November 2005 to revoke the patent Forgent Networks Inc. (Nasdaq: *FORG*) is widely asserting against the Joint Photographic Experts Group (JPEG) international standard for the electronic sharing of photo-quality images. In its filing, PUBPAT submitted previously unseen prior art showing that the patent, which was issued in 1987 to Forgent's subsidiary Compression Labs Inc., was not new and, as such, should be revoked. The PTO granted PUBPAT's request in February 2006 and rejected the broadest claims of the patent in May 2006. In November 2006, Forgent abandoned all assertion of the patent.'

with national offices. The US re-examination system appears to work reasonably well within the software field – an early victim was that of Compton's multimedia patent (US 5,241,671)[72] – and it may be, given there is centralisation of examination within Europe, that there should also be the development of a centralised system of re-examination if prior art should be found outwith the opposition period. Re-examination on novelty, for obviousness, sufficiency, etc. would also be a possibility but the procedural mechanism whereby this was initiated by outside parties appears more difficult to imagine, since it might effectively remove validity hearings from national courts – whether either the EPO or national courts would welcome that is a moot point.

There are other approaches being suggested. For example, opponents of the CII Directive have developed more confidence in their ability to challenge the EPO's hegemony. One of these is to 'privatise' examination, so that:

> The system of obligatory patent examination should be replaced by a system where patents are registered free of charge on the Internet, and examination is performed by private parties who, when they find an invalid patent claim, are entitled to charge an examination fee from the patentee.[73]

The idea behind such schemes is that those who produce 'bad patents' should pay for their negative costs to the community. However, even if there was a mechanism whereby all could agree that a specific patent was actually an invalid patent (surely the owners would not agree?), given the time taken to examine a patent to a sufficient degree of quality and the numbers of software patents which are being granted by the various offices, it seems unlikely that there would be sufficient goodwill in the community at large to examine all patents and one might suggest that even more such 'invalid' patents would enter the system.

Conclusion: does a European examination matter?

The whole point behind the setting up of the European Patent Office was that it would reconcile the problems of examination across Europe and provide a harmonised level of patent grant throughout the European economic community. It has certainly gone a very long way

[72] Though there are some who believe that the result was biased due to re-examination being requested by the PTO Commissioner rather than by an independent person. See T. S Hughes, 'Patent Re-examination and the PTO: Compton's Patent Invalidated at the Commissioner's Request', *John Marshall J. of Comp. & Info. Law* XIV (2) (1996).

[73] At http://a2e.de/ffii/epla/cpedu/exam/.

to achieving this aim, primarily through the very large numbers of examinations which it carries out in comparison with the national offices. However, it is clear that we have now – particularly in the software arena – arrived at a situation where the European examination is perhaps not sufficiently geographically wide-ranging to suit the software industry. Why is this? Principally because many of the software inventions being granted are based on telecommunications ideas which transcend both national and continental borders. Thus a high quality but slow examination in Munich is of little value when a low quality but speedy examination in, say, the US office leads to a situation where industry is faced with poorly examined multiple patents in software technology fields across the world. Although patents are national, the software marketplace is international: could Blackberry have continued as a successful enterprise if the European patents had been denied or revoked and the US ones stood and the US patent owner who claimed infringement by Blackberry refused a licence?

International pressure has therefore been applied to attempt to upgrade examination throughout the world, with WIPO and its Standing Committee on the Law of Patents being at the forefront. In February 2005 this committee:

> agreed that the following six issues should be addressed in an accelerated manner within WIPO with a view to progressive development and codification of international intellectual property law: prior art, grace period, novelty, inventive step, sufficiency of disclosure and genetic resources. These issues should be addressed in parallel, accelerated processes.[74]

Clearly, the attempt was being made to resolve various issues which are at the heart of examination but which – to the participants – were being held up by the attempt to produce a fully developed Patent Law Treaty. Unfortunately, a number of countries[75] objected to the attempt to push some aspects of the PLT forward at the expense of others and the acceleration endeavour failed. The result is that harmonisation across the technologised world continues to be fragmented and with differing examination goals and environments.

Given this, does the quality of European examination matter? My own view is that it does, and that the quality of examination is an important element in building confidence in the system. We see that a constant and underlying criticism of business method and software patents granted by the USPTO lies in the belief that so many of these are invalid and have been poorly examined. Examination is – as I have outlined in this

[74] WIPO, SCP/11/3, 7 March 2005. [75] WIPO, SCP/11/4, 21 April 2005.

chapter – difficult, but it does not seem to me to be impossible to produce a system of examination which is perceived by the users of the system to be as effective as current examination in chemistry, electronics, etc. The EPO system was always viewed as offering a Rolls Royce version of examination: in order to achieve this quality, it seems essential that the specific problems of examining software should be openly discussed and resolved with the communities who are affected by it.

5 Holding the line: algorithms, business methods and other computing ogres

The feature of using technical means for a purely non-technical purpose and/or for processing purely non-technical information does not necessarily confer technical character to any such individual steps of use or to the method as a whole: in fact, any activity in the non-technical branches of human culture involves physical entities and uses, to a greater or lesser extent, technical means.[1]

[With] applications for computer systems for the performance of business methods, therefore, it is essential to give careful and imaginative attention to the question of whether or not it is possible to identify some aspect of the system which can be said to provide a technical effect.[2]

Introduction

If the history of the EPC in its first two decades was that of continual attempts to overcome the software exclusions, then the current developments relating to business methods and algorithms offer a similar future: pushing the conceptual barriers of patentability until they finally stretch too far and break under the force of their lack of coherence. The power to do this lies with the patent attorney profession, who are 'repeat players'[3] in the process, with a relatively common goal: responding to their client's demands to extend the protection available for their inventions. The effect upon any examiner trying to 'hold a line' which is based on an abstract ideal or definition must be difficult, requiring both strong conceptual and organisational mechanisms. In fact we find that once a line has been so broken the results can surprise: patenting in sports fields in

[1] *Pension Benefits* (2000) T0931/95. All clear then?

[2] K. Beresford, *Patenting Software under the European Patent Convention* (London: Sweet and Maxwell, 2000), p. 183.

[3] This is an important concept from the sociological of study of law, suggesting that the strategic expertise built over multiple legal encounters is of great value. See M. Galanter, 'Why the "Haves" Come Out Ahead: Speculations on the Limits of Legal Change', *Law & Society Review* 9 (1974), 95–160.

the US where, for example, a means of holding a golf club is subject to protection.[4] 'Inventions' in 'sports technology' are not new: for example, we have seen methods to increase performance similar to the 'invention' of the Fosbury flop in high jumping which enabled an Olympic gold medal, or the skating technique in cross-country skiing which similarly enabled better competitive performance. Since 'usefulness' exists with this type of invention, the policy argument against them is usually that they remove the presumed 'level playing field' at the heart of competition. Given the huge potential income involved in many individual and team sports (supported by the intellectual property system and its associated broadcast and personality rights), it seems churlish to argue that the creators of new techniques should not be rewarded when users of their techniques can garner fabulous reward – but many European patent commentators do take the role of churl.

More seriously, since the strategic goal of business is to deny opportunity to the competition – by better goods, better salesmanship, better prices, dominating the technology or simply buying up the competition – the rise of internet commerce and the possibility of patenting gives another possible mechanism: excluding the competition through protection of business methods. If this is deemed unwelcome, can it be stopped? Or is there a slippery slope from software patent to business method and beyond? The evidence from the US is that the push for protection continually grows with, for example, the suggestion that plots and storylines, too, could be protected by patent.[5] This could hardly have been what the court in *State Street* expected when they created the 'useful' hurdle. In Europe, being able to draw a *clear* line in the patent sand is obviously advantageous since – as the US experience shows – there will always be those who see commercial advantage in pushing against existing limitations; if a clear line is feasible, it will be much easier to hold.

The conceptual apparatus underlying Art. 52 is not the first such entity to be subjected to stretching and manipulation. We could suggest that all mathematical concepts, too, undergo this process as the community tries to find proofs and refutations of those proofs.[6] In mathematics, such

[4] US 5,616,089, cited by D. Bambauer, 'Legal Responses to the Challenges of Sports Patents', *Harvard Journal of Law and Technology*, 18(2) (2005). Application EP1452208 relates to 'Improved golf club shaft and method of gripping golf club'. 'An improved method for holding a golf club, conventional or as taught herein, to provide additional stability to a golfer during a golf swing is also taught.' Filed January 2004.

[5] A. F. Knight, 'A Potentially New IP: Storyline Patents', *J. Pat. & Trademark Off. Society* 859 (2004). A critical reply is Anonymous, 'Pure Fiction: The Attempt to Patent Plot', *Harvard Journal of Law and Technology* 19(1) (2004), 231–52.

[6] See I. Lakatos, *Proofs and Refutations* (Cambridge: Cambridge University Press, 1976).

developments are usually positive, with the growth of the field and understanding. Thus the very attempt to produce definitional axioms produces counter-examples which require monster-barring strategies to ignore these or refinements and/or amendments to the axioms to include the counter-examples. Definitional conflict is thus – eventually – welcomed as a source of advancement.[7] An interpretive approach to law would suggest that legal rules, too, are amenable to such manoeuvring but, with respect to Art. 52, not all view such developments – as we have seen in Chapter 3 – as positive.

In discussion of patent protection for software, the practical questions posed are: whether it is useful – in policy terms – to hold the current Art. 52 interpretation; and whether it is possible to hold it. This chapter will therefore focus upon the following questions:

- Where is the current line drawn with regard to 'traditional software' at present?
- Is mathematics protectable and/or should it be?
- Where might the line relating to protection of algorithms 'as such' be drawn?
- Should/could there be a line between traditional and business method software artefacts?
- And what about 'information' – the raw material of information technology?

Conveniently, some of these points have been discussed in the conjoined *Aerotel* and *Macrossan* appeals in the UK, where the border between traditional software and business method software arose. The related point of what might be the future for the institutional structures which uphold these lines will be dealt with in the final chapter.

The thrust of this chapter is that there are many solid reasons for allowing protection of software as a technology 'as such'. In Chapter 1 we outlined the manner in which software was viewed as 'mere data processing' unless it was linked in with some hardware to produce a hardware-like device, and in Chapter 2 we outlined why 'invention' was an appropriate aspect of software. In this chapter we focus on reasons why the current position is an artifice and where the fault lines lie – those characteristics which will (if the prediction from Chapter 1 is correct) force realignment and allow protection for software as software. In the final chapter, we look at the possible

[7] Though there are still many around with neo-logical positivist approaches to mathematics. Many, indeed, have moved into computer science where they undertake such exercises as trying to prove programs correct (i.e. debugging through mathematical logic). This seems impossible to this author, given the 'cogs and wheels'/Albion mills metaphor for the construction of virtual objects earlier in this text.

paths towards this re-alignment and also suggest that, rather than being a totally negative development for computing, it may begin to offer computing a sounder theoretical basis than it presently has.

What is currently protectable as 'software'?

(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1: (c) . . . programs for computers.

My interpretation of the current situation is that the board of appeal IBM decisions potentially radicalised protection: it brought the programmer's perspective to the fore (albeit in a kind of 'furtive foreground', if that is possible[8]). The two accepted software claims in the IBM applications which moved the invention to software 'as such' freed the invention from a *Vicom* link to hardware – from requiring through a combination of hardware and software an effect external to the program to only requiring the internal effect of the software itself. Writing as late as 2004, Lloyd was describing the effects of the IBM decisions as significant, and – though seemingly supportive of the developments, describing the failure to progress as an abdication of responsibility – suggested that these should have been made in a more political context.[9]

There was, then, certainly optimism amongst the legal commentators who understood the logical incoherence of the pre-IBM decisions. Yet, when we view what has happened since these decisions, we must conclude that they have not really had as much effect as might have been imagined. In fact, the acceptance of software claims – 'programs as such' – was only partial. Since the model of acceptable invention was still based upon notions of machine, 'pure' software claims still have to have that special relationship with the machine. Currently, protected software is protected 'as such', but not all software is given that protection: the examiner must be persuaded that the software is machine oriented (or in the terminology of the boards of appeal – 'technical'). We are thus still at that point which Banks noted:

> Whether or not there can be said to be a real distinction between a program invention claimed in this rather roundabout way and a claim to the program itself is at least open to doubt.[10]

[8] That is, commensurate with the public statements that 'programs as such' were still not being protected – i.e. that the Board of Appeal had not really legislated despite what it looked like.

[9] I. J. Lloyd, *Information Technology Law*, 4th edn (Oxford: Oxford University Press, 2004), p. 439.

[10] Para. 474. M. A. L. Banks (Chairman), *The British Patent System: Report of the Committee to Examine the Patent System and Patent Law* (London: Stationery Office, 1970), Cmnd. 4407.

This position remains as confusing today as it did to Banks in 1970 and we find the judiciary still wrestling to find a more formal approach than the intuitive 'engineer seeking for the technical'. Thus, in *CFPH's Application*, the judge attempted the use of a metaphorical 'little man' to try to highlight the line across which patentability could not be offered. This 'little man' test might thus help to distinguish when an application dealt with a 'program as such' or with an artefact which could be patented:

> But the mere fact that a claimed artefact includes a computer program, or that a claimed process uses a computer program, does not establish, in and of itself, that the patent would foreclose the use of a computer program. There are many artefacts that operate under computer control (e.g. the automatic pilot of an aircraft) and there are many industrial processes that operate under computer control (e.g. making canned soup). A better way of doing those things ought, in principle, to be patentable. The question to ask should be: is it (the artefact or process) new and non-obvious merely *because* there is a computer program? Or would it still be new and non-obvious in principle even if the same decisions and commands could somehow be taken and issued by a little man at a control panel, operating under the same rules? For if the answer to the latter question is 'Yes' it becomes apparent that the computer program is merely a tool, and the invention is not about computer programming at all. It is about better rules for governing an automatic pilot or better rules for conducting the manufacture of canned soup.[11]

But the 'little man' test is, once again, 'machine-oriented': if the application can be viewed as a machine 'in the round' rather than as a method for carrying out a task which is based in a program alone, then it is protectable. This includes viewing it as some new and inventive set of rules to be followed by a 'little man at a control panel' (indicating, one supposes, that he is simply a cog in the machine) then it is similarly protectable. The 'little man' test at first seems sensible because it appears to provide a simple technique to differentiate patentable from non-patentable inventions. It also applies a similar kind of test to patentability of computer related inventions from those applied in other areas: would we remove the possibility of patentability from a chemical process because it could be carried out by a little man at a laboratory desk? Unfortunately, in practice, the test seems less easy to utilise that we might imagine, as the discussion in Oracle's patent application demonstrated.[12] For example, does it mean that we should not be too concerned about an invention

[11] *Patent Applications by CFPH LLC* [2005] EWHC 1589 (Pat), para. 104.

[12] BL O/254/05. The application related to conversion of, for example, SGML to HTML formats. Generally, this kind of conversion is not always easy and invention would be valuable: the Department of Constitutional Affairs' Statute Database project in the UK has had significant problems in moving from the early SGML version to XML (an extended version of HML) and a system which could have carried this out with minimal human input might have saved the taxpayer large sums of money.

having a basis in software, or is there something we need to look at in terms of the role of the little man, or does it just mean that if there is no equivalent to the aircraft or soup then there's no patentable invention? As with many judgments dealing with patentability of software, we can be left as confused after the decision as we were before,[13] yet a patent office must include such directions within their guidelines.

The continuing problem, for the examiner and judge, is determining just where the line dividing a 'technical' contribution and a 'non-technical' contribution lies. The guidelines for the EPO (and the slightly differing guidelines for other offices such as the UK[14]) try to provide a simple, step-by-step approach, but in practice (as we saw in Chapter 4), the result may be based as much in the nature of the interaction between examiner and applicant's agent as in the formal procedure to be followed.

So where are we with protection? Basically, not much further on from that of *Vicom* – the programmer's voice is still in reality little heard. However, if the programmer's words are moulded (by the patent attorney) to produce a machine-like artefact or process then protection for 'software as such' is available. The disadvantage of this – in the author's view – is that software technology is not being dealt with in the manner properly required by such an important technology (i.e. with proper development of claim terminology, sufficiency, clarity, etc.) but with a descriptive language borrowed from a different technology. However, the advantage which arises from the current position is to those who design or manufacture inventive artefacts which include a program (for example, DVD players and other computerised devices). They have protection for the device as a whole and, if the invention lies in the software, then that software is protectable, too. But without the linkage with the device, there is no protection.

The situation at present appears to be that since a useful invention in software cannot be protected 'as such', it must be tied to some other device or technical process or business system. This implies that we can have a large number of inventions which can be protected (application X using inventive technique A, application Y using inventive technique A, application Z using inventive technique A ...) but the primary useful invention – technique A – cannot be similarly protected. Such a situation appears illogical.

[13] That appeared to be the common perception after the earlier Fujitsu case – see I. Lloyd, 'Software Patents After Fujitsu. New Directions or (another) Missed Opportunity?', *Journal of Information, Law and Technology* 2(1997). He continues the critical stance on *Fujitsu* in Lloyd *Information Technology Law*.

[14] Patents Act 1977: Patentable subject matter, 2 November 2006.

It is important to see that the line as currently drawn is not a technical one at all. It is, in fact, a social line which is being drawn – the Boards of Appeal are representing an engineering community and their view of what is technical is one which is relevant to the intellectual model of that engineering community. We see this in the *Pension Benefits* quotation at the start of this chapter: they cannot use *technical* as a dividing line because technicality is everywhere, so they draw from a shared community of experience and thought – the artefact (a machine) is central to that experience and so it is that which they use as a line in the sand. Unfortunately, socially constructed lines are difficult to hold.

Algorithms and mathematics: are the models broken?

(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1:

(a) discoveries, scientific theories and mathematical methods;

. . .

(c) schemes, rules and methods for performing mental acts . . .

Algorithms – as step-by-step procedural operations – are not specifically mentioned and excluded in Art. 52. There have been two means, however, through which exclusion could be enabled – as mathematical methods on one hand or, on the other, as rules and/or methods for performing mental acts. The distinction between the two is artificial, since any step-defined algorithm can be modelled mathematically and many computer-implemented mathematical methods are specified as a sequence of steps or rules.

Vicom[15] was the first of the applications which were taken to appeal at the EPO and which dealt with the application of algorithm. The objection of the examiner was that the processing of an electrical signal was computer-related and also involved a mathematical method – a means of cleaning up a signal by applying mathematical filters to a data structure. Clearly this is true, but the Board of Appeal took the view that:

> 6. The Board, therefore, is of the opinion that even if the idea underlying an invention may be considered to reside in a mathematical method a claim directed to a technical process in which the method is used does not seek protection for the mathematical method as such.

The application was sent back for re-examination, and although the office intended to proceed to grant,[16] formalities were not carried through by the

[15] *Vicom Systems Inc's Application* [1987] 2 EPOP 74; T0208/84.
[16] Letter from EPO to applicant, 4 August 1987.

applicant and the application was deemed to have been withdrawn.[17] However, *Vicom* set in motion the philosophy laid out in Chapter 1 – that so long as the invention was machine-like, then it was protectable. We thus commonly have applications which routinely incorporate mathematical operations or algorithms, and which proceed to successful grant. For example, the early EP0138243 outlined a method of increasing the height of characters on a computer display was filed in 1984 and granted in 1987. Many patents also involve computations which are used in cryptography: for example, EP1400056 (Cryptographic Authentication Process, granted 2004) which does not mention a machine in the claims but which has obviously been read as being used in computer processing.

The question is, why should protection be given to someone who finds a specific use for a mathematical method (in a machine application), but not to the originator of the method itself despite the fact that the method will be known to be generally useful? Why is this dividing line here when it is not obvious in the mathematical technology that there is any real distinction between the reasoning or expertise applied? For example, we could produce a method which effectively allowed us to carry out rearranging work (permutations) on data – say computing inverses of a series in a program which carried out sorting. This is obviously interesting work but why is it viewed as protectable (when reframed to satisfy patent requirements) when the original work upon which it is based would not be protectable?[18] Is there any real difference between the mathematical usefulness of a method and a method applied where it was designed to be applied?

While there is a very strong antipathy amongst some of the academic mathematical community to the idea of protection of mathematical inventions by patent, this is often balanced by the understanding that mathematical expertise has an economic value. Knuth, for example, suggested that elements such as *pi* should not be protected, but with something non-trivial 'like the interior point method for linear programming' there was more justification for protection because it made it public rather than being kept as a trade secret[19] – a tactic which had been used by

[17] Letter from EPO to applicant, 6 October 1988. A US patent was issued – US 4,330,833.

[18] D. Knuth, *Fundamental Algorithms*, Vol. 1, *The Art of Computer Programming*, 2nd edn, (Reading, MA: Addison Wesley, 1973), p. 179 outlines this and then notes the origin in Victorian mathematical methods.

[19] 'Thus, we have reached the end of the era when the joy of discovering a new algorithm was satisfaction enough! Since programming is strongly analogous to the fabrication of a machine, and since computer programs are now worth money, patented algorithms are inevitable. Of course the specter of people keeping new techniques completely secret is far worse than the public appearance of algorithms which are proprietary for a limited time'. D. Knuth, *Sorting and Searching*, vol. 3, *The Art of Computer Programming* p. 319.

Isaac Newton to protect his invention of calculus.[20] He pointed out that while he did not charge a penny every time others used a theorem which he had proved, he did charge to tell them which theorem should be used.[21] Knuth also wondered whether a large integer which proved mathematically useful might be patentable or whether rather than being man-made it was actually God-given. A reply came several weeks later pointing out that such a large integer had indeed been given protection.[22]

Importantly, as outlined earlier, mathematics and algorithms exist in a reasonably complex relationship with each other, where theories, models and code interact: a factor which lawyers sometimes fail adequately to grasp. Thus, in Newell's well-known paper, *The Models are Broken . . .*,[23] he pointed out, in reply to Chisum's argument that there was a solution in legal interpretation to the problem of algorithms,[24] that he was somewhat bedevilled by confusion about the basic conceptual models used by lawyers. Newell's paper argues that law requires a stable view of the algorithmic process whilst computer science thrives on change, suggesting that never the twain shall meet. It is possible that many legal readers of this paper will not really have understood why Newell was making this kind of claim that law could not satisfactorily cope with the new technology: in fact it was because, as he went on to say, his career was aimed single-mindedly at understanding the human mind.[25] He was certainly involved with computer science and mathematics, but he viewed algorithms as central to the working of the brain, hence his work with Simon on *Human Problem Solving*.[26] Someone who viewed algorithmic rule-based processing as the manner in which humans reason and solve problems would hardly be likely to accept Chisum's more traditional perspective.

[20] Is calculus an invention rather than a discovery? Many mathematicians, I suspect, would consider it to be an invention. Newton argued with Leibniz over who was the true inventor both having concurrently arrived at the same techniques.

[21] P. 324. *Notices of the AMS*, vol. 49, No. 3.

[22] 'When asked about software patents, Donald Knuth . . . conjectured that it might be possible to patent a 300-digit integer. The readers might be interested to know that I have already patented a 309-digit integer as claim 37 of US Patent 5,373,560, issued in 1994. It relates to cryptography, and it is not as interesting as what Knuth had in mind. At the time, some people thought that patenting a number was a new extreme in silly software patents, but now we have business method patents that are even sillier.' Roger Schlafly (Received 13 February 2002), *Notices Of The AMS* (2002) p. 543.

[23] A. Newell, 'The Models are Broken, the Models are Broken!', *U. Pitt, L. Rev.* 47 (1986), 1023.

[24] D. Chisum, 'The Patentability of Algorithms', *U. Pitt, L. Rev.* 47 (1986), 959.

[25] Cited by his colleague, Herbert A. Simon, in a memoir at http://stills.nap.edu/readingroom/books/biomems/anewell.htm.

[26] A. Newell and H. A. Simon, *Human Problem Solving* (Englewood Cliffs, NJ: Prentice-Hall, 1972).

Another aspect which is often ignored in debates about protection for mathematical algorithms is that many of these methods are intrinsically related to the computer and only arose as part of the development of the computer. A reader of Morris Kline's classic *Mathematics in Western Culture* (first published in 1953)[27] will not find the term algorithm in the index and there is almost no reference to algorithmic procedures. A text on 'Mathematics and Western Culture' published today without discussion of algorithms would be an object of reviewer ridicule, since algorithms have become so central to many important areas of mathematics (which areas were relatively minor until the 1950s). Take numerical analysis – a discipline which, while it certainly has a long history, has become enormously important with the use of computation. Mina Rees, from the Computing Program at the office of Naval Research was one of the first to push this new field. She wrote:

> [Henry S. Tropp] commented that, as a young assistant professor of mathematics in 1959, he had never even heard of numerical analysis as an independent area of mathematics. He said, '[one] did applied things in various courses and [one] had a course called applied mathematics for physicists and engineers which was more applied differential equations ... And [one] suddenly realize[d] that out of this wartime environment a new independent branch of mathematics had suddenly blossomed. The fact that you founded an Institute for Numerical Analysis indicates an early realization of the independent area of research, where it really didn't exist before.' I replied, 'But computers didn't [exist] either.[28]

This kind of approach – which parallels the developments of hardware to improve speed and efficiency – is surely as important as the development of hardware models. During this author's period in a Mathematics Faculty,[29] I shared an office with a colleague who undertook numerical analysis-based research into fluid mixing in a paint gun. He could not, he told me, tell whether his mathematics were correct, his program was correct or his algorithm from which the program was derived was correct – the problem was so complicated that errors could be found in any part. These kinds of tasks are immensely *practical* and yet, to the patent system, appear less worthy of reward than a particular design of a paint gun which may have derived from this work.

The irony of there being practicality in something which is as seemingly abstract as an algorithm is accepted by some mathematicians who are, concurrently, opposed to patent protection for mathematical algorithms.

[27] M. Kline, *Mathematics in Western Culture* (New York: Oxford University Press, 1953).
[28] M. Rees, 'The computing program of the Office of Naval Research, 1946–1953', *Communications of the ACM* 30(10) (1987).
[29] I am not a mathematician – computing is often found in such faculties.

For example, Klemens, in *Math You Can't Use*,[30] argues that software and mathematics are identical. In purely theoretical terms this is true, but in practice the ways in which a programmer constructs a program are rarely similar to that of the mathematician – the programmer is building a virtual world of objects which can be manipulated and pays little attention to computational or other theory. Klemens's argument is that, although mathematics and software are identical, maths should somehow be divided off from software, in order that mathematicians can utilise what he suggests are 'pure mathematical algorithms' without fear of falling foul of the patent system. His solution to the problem is very much like that of the EPO: that is, if it is a physical device it can be protected but, if not, then no protection is offered. Klemens uses the notion of a 'physical state machine', which derives from the theory of computability. Essentially, this is a physical machine which can operate on inputs and outputs (i.e. a computer or a processor chip): 'an inventive physical state machine may be heavily informed by mathematics but would make a nontrivial extension of that mathematics into the physical world.'[31] This is special pleading – Klemens is aware that the inventive activity is in the mathematics but wants to keep that inventive activity free to use, but still provide ownership rights somewhere along the line. He therefore gives protection to the manufacturer rather than to the inventor (since the manufacturer can produce an 'application X using the inventor's technique A' without problem). This is fine as a policy decision, but does not appear to accord with the rhetoric of patenting and the reward theory – why should more economic advantage accrue to the user of someone else's mathematical enterprise?

A different approach using standard patent law could in fact be used – the experimental exemption, for example – if it is the right to utilise a protected algorithm in research which Klemens wishes. This exemption allows what Rimmer has called 'the right to tinker'[32] and although it is primarily of economic importance in the chemistry area, it could well cover mathematical usage.

Algorithms which have been protected in the past appear to have been awaiting the development of the computer. One example is that of the Soundex algorithm, which provides a method for computing a word which sounds like another – important in misspelling words and names

[30] B. Klemens, *Math You Can't Use: Patents, Copyright, and Software* (Washington DC: Brookings Institution Press, 2006).
[31] Klemens, *Math you Can't use*, p. 65.
[32] M. Rimmer, 'The Freedom to Tinker: Patent Law and Experimental use', *Expert Opinion on Therapeutic Patents*, Vol. 15, No. 2, 1 February, pp. 167–200 (34).

for example. This was protected in 1918 and 1922[33] as an advance in indexes (that is, in 'device' format, but the algorithm was central to the device) and was particularly useful in processing of census data at a time when a large number of foreign names might be misspelt or misheard. And algorithms have been protected in hardware form since the beginning of computing – the bubble sort sorting technique was protected in hardware form in US 3,029,413 and US 3,034,102. As with the practice today, both of these inventions lie in the algorithm, with the device being dressing: form is more important than substance.

Since protection for algorithms has been around for some considerable time, it is not too obvious that developments in computing or mathematics have been held up in any way. The opponents of protection consistently push the idea that creativity will be stalled should software or algorithms be patentable; yet, clearly if, as is the case, software has been patented for some time, there still appears to be no end in sight to development and progress. There may be problems – monopoly issues with patent pools, over-broad claiming, etc. – but these have not produced chaos in the system.

But what of the line which we may wish to draw? If algorithms are to be protectable, should those which are not technologically based also be protected? Can they be prevented from gaining protection? For example, it is common to use diagrammatic representations of algorithms to aid procedures in all kinds of areas. The UK Health Protection Agency utilises an algorithm in flowchart form to advise on the management of returning travellers from countries with avian flu.[34] Assuming this is a novel and non-obvious method of management, should it be protectable? It might be excluded under the medical exclusion in Art. 52(4)[35] or even as presentation of information in Art. 52 (2)(d),[36] but should it also be excluded as a rule or method for performing a mental act? We might believe it is not technology, but others might suggest that it is an example of 'nursing technology' and so the algorithm is essentially technical. Would that argument get past an examiner?

The Boards of Appeal of the EPO have circumvented the problem of defining the term 'technology' by using an engineering *notion* of technology and, to engineers, this nursing use is obviously a rule or a mental act,

[33] US 1,261,167 and US 1,435,663.

[34] www.hpa.org.uk/infections/topics_az/influenza/avian/algorithm.htm.

[35] EPC Art. 52(4): 'Methods for treatment of the human or animal body by surgery or therapy and diagnostic methods practised on the human or animal body shall not be regarded as inventions which are susceptible of industrial application . . .'

[36] '(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1: . . . (d) presentations of information.'

so would be unlikely to gain protection. This is a perfectly reasonable way to go forward: however, such an intuitive approach to what is technical both leaves a valid field of technology (computer science) outwith the definition and potentially includes applications which might be viewed as invalid fields, simply because they can be forced into the 'technical effect' straight-jacket. Thus, Beresford's challenge to his patent attorney colleagues is clear in the quotation at the beginning of this chapter: use 'imaginative attention' and take on the examiners, he says, and undermine the line they are trying to hold by using their own definition of technology.

Business methods

(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1: (c) schemes, rules and methods for . . . doing business.

It is clear that significant application numbers are building in the business method (G06Q) field. In a short two-week period in November 2006, some thirty applications were published covering a wide area of 'invention':

- Patient Record System (EP1721280).
- Data Processing Method for Time Optimal Computation of Large Result Data Sets (EP1722326).
- Retail Marketing Method (EP1721291).
- A Multi Site Solution for Securities Trading (EP1721253).
- Method of Interactive Advertising (EP1721460).
- Financial Transactions with Messaging and Receipt Charges (EP1721292).
- Method for Selecting A Portable Device With An Electronic Tag (EP1719299).
- Electronic Trading System (EP1719074).
- Dynamic Pari-Mutuel Market (EP1719076).
- Multimedia Content Distribution Platform and Procedure To Create Such A Platform (EP1720126).
- A System and Method for Processing Audit Records (EP1719065).
- Method and Apparatus for a Collaborative Interaction Network (EP1716504).
- Computer System and Computer Implemented Method for The Management of a Bonus System (EP1717738).
- A Device and a Method for Using an Enhanced High Priority Calendar Event (EP1716526).
- Methods and Systems Rating Associated Members in a Social Network (EP1716533).

- System and Method for Developing and Implementing Business Process Support Systems (EP1716528).
- System and Method Employing Radio Frequency Identification in Merchandising Management (EP1716518).
- Method and System for Word of Mouth Advertising via a Communications Network (EP1716525).
- Method for Ordering a Product At An Online Shop Connected to a Communication Network (EP1717748).
- Method for Broadcasting Information in a Cellular Telephone System, and Device Therefor (EP1717752).
- Systems and Methods for Software Development (EP1716483).
- Method and Device for Recording of Data (EP1716527).
- Filtering Process for Instant Messaging (EP1717739).
- Information System (EP1716529).
- Method and System for Conducting An On-Line Survey (EP1716534).
- E-Mail Type Electronic Message Buffer Memory Device (EP1716532).
- Method of Optimizing Freight of Goods (EP1716523).
- Electronic Trading System (EP1716530).
- Standardizing Intelligent Mail Processing (EP1716524).
- A Project Management Method and System (EP1716509).

It appears to be the case that someone with business technology interests would glean more about current developments in their field from a perusal of patent applications than a computer scientist would learn about their own field: the latter inventions are hidden by confusing classifications, and by being enveloped within a technical device. On the contrary, here we see that business methods almost proudly fly the flag of their 'innovation'. The EPO has been prepared to grant such business methods – the infamous '1-click' Amazon patent (EP0927945 – 'Method and system for placing a purchase order via a communications network') was allowed to proceed to grant in 2003. A number of oppositions were filed and a robust reply was made to these from the patentee's representative,[37] but there appears at the time of writing to have been no formal conclusion to this opposition. Claim 1 of the patent is:

A method in a computer system for co-ordinating delivery of a gift from a gift giver to a recipient, the gift and recipient being specified in a gift order, the method comprising:

[37] Letter to EPO, August 2004.

- determining whether the gift order includes sufficient information so that the gift can be delivered to the recipient;
- when sufficient information is not provided in the gift order, obtaining delivery information from one or more information sources; and
- when sufficient delivery information can be obtained from the additional information sources so that the gift can be delivered to the recipient, directing the gift to be sent to the recipient as indicated by the deliver information.

Whether this is a 'device-like' method is open to argument, but clearly shows the application types which are building up in the EPO and other patent offices and in some cases moving on to grant. A similar approach to the EPO was taken in Australia, where a business method must arrive at a 'useful product' – a physical phenomenon or effect resulting from the working of the method (more than mere 'intellectual information'). Interestingly, for the Macrossan application discussed in Chapter 4, the court in *Grant*[38] suggested that:

> 34 The interpretation and application of the law would not be considered as having, in the words of *NRDC*, an industrial or commercial or trading character, although without doubt it is an area of economic importance (as are the fine arts). The practice of the law requires, amongst other things, ingenuity and imagination which may produce new kinds of transactions or litigation arguments which could well warrant the description of discoveries. But they are not inventions. Legal advices, schemes, arguments and the like are not a manner of manufacture.

Macrossan's application was granted in Australia and involves the interpretation of law and procedure to provide a system which effectively replaces a lawyer in practice. Is Macrossan's Australian patent invalid after *Grant*? It is difficult to tell because the border between where legal practice begins and commerce ends is not clear to this author: I tell law students that they must remember that the practice of law is a business.[39]

Such problems in asertaining where the line exists bedevil the examination of many patent applications. In Hitachi[40] the board of appeal were clearly unhappy about the basis of the application being essentially a Dutch auction, where the technical problem (having delays caused by the communications network running the auction) was overcome, or rather sidestepped, by changing the rules of the auction rather than by solving them by the technical means of, say, having an extra server somewhere. In the UK, such questions had been discussed in various judgments but a lack of clarity appeared to lead to two actions in the High

[38] *Grant* v. *Commissioner of Patents* [2006] FCAFC 120, 18 July 2006.
[39] See J. Morison and P. Leith, *The Barrister's World and the Nature of Law* (Buckingham: Open University Press, 1992).
[40] T258/03.

Court being considered useful tests for the Court of Appeal, who allowed the appeals to be heard. These were *Aerotel*[41] and *Macrossan*,[42] the first having been declared invalid and the second refused on appeal in the Patents Court. We have already outlined the nature of the *Macrossan* application in Chapter 4. The *Aerotel* patent had a similar business basis but involved communications technology: the involvement of a special exchange which, through a 'special code', allowed calls to be pre-paid and accessed through the special exchange. Jacob LJ, for the court, noted that there were method and system elements to the claims – a 'method of making a telephone call' (claim 1) and 'a telephone system for facilitating a telephone call . . .' (claim 2). With *Macrossan*, too, there was a method and system element: 'A method for producing documents for use in the formation of a corporate entity using a data processing system, the system comprising . . .'

To the computer scientist there appears little between the two applications, and in fact in the original *Aerotel* decision on validity, Lewison J had noted that none of the technology was new and that, despite the system having (in the words of the patent holder) revolutionised the way that telephone calls could be made, '[n]othing else in the patent of a technical nature, in the sense of equipment, is said to be new and none of that technical equipment is described except in the most general terms'. What was being described was being viewed by the judge as a non-technical business method, just as many computer scientists would similarly view the patent. Computer scientists would view the two as similar primarily because they do not actually rely upon any specific hardware elements – the inventive aspects would lie in the manner in which these were implemented and programmed. The patent and applications may both cite hardware but it is the control elements which are important (i.e. the software, algorithms, data/information structures, etc.) *Aerotel* used well-known digital communication exchange hardware but linked them together with software control (using codes, payment schemes, etc.) and *Macrossan* utilised well-known hardware (including the internet, and servers) but the invention, if any, was in the software system which produced the legal documents.

[41] *Aerotel Ltd.* v. *Telco Holdings Ltd* [2006] EWHC 997 (Pat). The patent is GB2171877: Telephone System ('A telephone system is provided for enabling prepayment for telephone calls, wherein special code and credit information is stored in memory in special exchanges and debited as the call progresses. The system includes a special exchange having a memory 86 for storing special subscriber codes and credit information. Verifying means 83, 84 verifies a caller code and checks that the caller has available credit. Means 89, 91 are provided for then connecting the caller to the called party.')

[42] *Macrossan* v. *Comptroller-General of Patents, Designs And Trade Marks 2005* [2006] EWHC 705 (Ch).

Aerotel succeeded and their patent was reinstated but *Macrossan* failed. Why was this? *Macrossan*'s 'method' it was held:

> is for the very business itself, the business of advising upon and creating appropriate company formation documents. . . . The final step is to ask whether there is anything technical about the contribution – there obviously is none beyond the mere fact of the running of a computer program.[43]

But *Aerotel's* use of hardware was deemed to provide a technical contribution:

> The important point to note is that the system as a whole is new. And it is new in itself, not merely because it is to be used for the business of selling phone calls. So, moving on to step two, the contribution is a new system. It is true that it could be implemented using conventional computers, *but the key to it is a new physical combination of hardware*. It seems to us clear that there is here more than just a method of doing business as such.[44]

This is in line with the thinking of the EPO, and the court in its discussion of the law tried to fit in with the European approach. But as Jacob LJ described it:

> It is clear that a whole range of approaches have been adopted over the years both by the EPO and national courts. Often they lead or would lead to the same result, but the reasoning varies. One is tempted to say that an Art. 52(2) exclusion is like an elephant: you know it when you see it, but you can't describe it in words. Actually we do not think that is right – there are likely to be real differences depending on what the right approach is. Billions (euros, pounds or dollars) turn on it.[45]

A critical reading of the *Aerotel* patent and the *Macrossan* application – which suggests that both are more business method than device – might propose that the only real difference was that one must have been reading the urgings of Beresford and his 'imaginative attention' to 'identify some aspect of the system which can be said to provide a technical effect'. That person does not appear to have been Macrossan. Macrossan put insufficient imagination into finding a technical contribution to give the necessary form to his application. The burden is certainly on the patent applicant to ensure that the claims fit the requirements of the examiner: and this appeal judgment now gives a relatively focused direction as to what must appear in the application document in order to pass the hurdle of technical contribution. Patent attorneys will be mulling over the ruling, and applying the required imagination to their draft applications – and the difficulty of knowing when a substantive rather than a formal advance is presented to the examiner will surely become progressively more difficult.

[43] *Aerotel Ltd* v. *Telco Holdings and others Rev 1* [2006] EWCA civ 1371 at paras. 71, 72.
[44] Para. 53 (emphasis added). [45] Para. 24

Another critical reading would suggest that we have a classic example of how, in attempting to prevent software patents 'as such' from gaining protection, we allow something which some might consider much worse – business methods. The gatekeeper which is supposed to deter software actually enables business methods to get protection more easily – simply by locating a sufficient technical contribution we turn a business method into a non-business method and system. These kinds of applications, the critic would suggest, are actually technically very easy to implement: once you know the business model, it could be implemented any number of ways by a person skilled in the art. But the EPO's approach hides that element and allows the protection as something new – an artefact which is primarily an 'inventive' business method implemented with standard technological knowledge. Such a conclusion is fine if the goal of the patent system is to protect computerised ways of doing business, but according to the rhetoric of the patent offices this is not their desired goal.

It may be that business methods are not are as problematic as their critics would suggest, a position taken by Stobbs, who argues that patents are but one part of a business strategy and not always the most important. Further, that:

> civilization as we know it did not end when the Patent Office granted Priceline.com its business model patent on the name-your-own-price concept . . . It did not give Priceline.com a broad monopoly over offer-and-acceptance.[46]

This is obviously a position which the anti-software lobby would oppose vehemently.

Information

The discussion in Chapter 2 of the nature of programming suggested that it was the building of virtual worlds using data objects and methods of manoeuvring those data objects. These virtual worlds reflected the world outside the program – either representing it in terms of information input/output or controlling other devices such as robotic arms. At the heart of the programming process is thus data, or *information*. My reading of the exemptions of Art. 52 suggest that it is the 'informational' of the

[46] G. A. Stobbs, *Business Method Patents* (New York: Aspen Law & Business, 2002) p. 22. Priceline.com was one of the ideas output by Walker Digital (http://www.walkerdigital.com/) a venture designed to 'invent' and protect various new computerised business models. At November 2006 the website of the company was laying claim to over 200 US patents. The difficulty of moving these from patented idea to successful business perhaps demonstrates Stobb's argument that patented business methods do not unduly affect the marketplace. Once again, many will not agree with this view, pointing out that unnecessary rent-seeking is being encouraged rather than inventive activity.

exemptions which underlies the general sense of them having a common underlying theme, rather than any 'abstract nature'. The problem for 'information' is that it is viewed as an abstract form: it is a collective term which almost denies that it can related to specifics, but it certainly is specific in terms of a programmer.

The exemption in Art. 52(2)(d) relating to 'presentations of information' is of different aspect, being about *presentation* rather than *processing*. The concern of programmers is not really presentation, but how to manipulate information in the form of data and data structure. Thus many of the inventions which have been granted protection and which are device-like are really inventions which process information.

We can see just what the difference between a programmer's view and a European patent attorney's view is by looking at the differences between an EP application and a US patent. For example, the early EP application EP0179350 (filed in 1985 by Wang)[47] entitled 'Data Processing System' presented a combination of hardware and software (together with the requisite diagrams showing that this was a device rather than software) and was derived from a similar family of application to the US Office including US 4,713,754 ('Data structure for a document processing system') whose abstract outlined what was the underlying (but hidden) software invention in the EP application:

> A data structure for use in a document processing system corresponds to a document comprised of one or more Pages. Each Page is subdivided into one or more non overlapping Areas, each Area in turn being comprised of one or more types of Layers. Each Layer type is expressive of a particular type of information such as text or graphics information. Different Layer types may be superimposed to comprise the contents of an Area.

The patent attorney can be seen here to be trying to mould a software invention into a European-style format which the then new *Vicom* decision required – a situation which occurs frequently, when the attorney is given the opportunity to do so (rather than simply acting as a post-box for the European applications). The original was the invention of a data structure to handle word processing documents, but this became a mangled device-like entity by the time it reached the European Office. The programmer's view was hidden behind an engineer's notion of 'technical'. This was published in 1986 – so the practice of moulding software into hardware form has been obvious for twenty years or more, and the examiners would hardly fail to have noted from the patent family that the inventive aspect of the application, if any, was software.

[47] This was withdrawn on 1 July 1988. Examination dossier was destroyed 22 June 1993.

Unfortunately – or fortunately, depending upon one's view on the desirability of software patents *as such* – it has been easier to exclude programs which inventively handle information than those which implement business methods. This has particularly been the case in the UK where, as Lloyd has suggested:

> In general, applications relating to software have fared less well before the UK courts. In more than half of the cases brought before the European Patent Office Board of Appeal, applications have been declared patentable. Of six reported cases brought before the UK authorities in only [*Gales Application*] was an application declared to constitute patentable subject matter, a determination at first instance, which was reversed before the Court of Appeal.[48]

We continue in a position where the programmer's inventivity is protectable, but only when it is conceived to have been developed by someone other than a programmer – the person 'skilled in the art' cannot, in the current situation, be a 'mere data processing expert'. The programmer does not really care whether the data structures and algorithms he is working with actually change data and information or change the physical state of a machine. They are all, in terms of programming, simply manipulations of virtual objects. As Judge Newman pointed out in her dissent in *In re Schrader*:[49]

> The majority now imposes fresh uncertainty on the sorts of inventions that will meet the majority's requirements. All mathematical algorithms transform data, and thus serve as a process to convert initial conditions or inputs into solutions or outputs, through transformation of information. Data representing bid prices for parcels of land do not differ, in section 101 substance, from data representing electrocardiogram signals (*Arrhythmia*) or parameters in a process for curing rubber (*Diehr*). All of these processes are employed in technologically useful arts.

Newman is correct in her assertion that that data representing financial objects is no different from electrical signals. But, tied in to the thinking of Board of Appeal 3.5.1 in *Vicom*, no-one in Europe appears to be able to dig themselves out of the logical hole which is the conclusion of *Vicom's* technical effect approach.

Conclusion

There is a realisation amongst commentators that a logical contradiction exists. Unfortunately, the political will which appears necessary to resolve

[48] I. J. Lloyd, *Information Technology Law*, 4th edn (Oxford: Oxford University Press, 2004) p. 410.
[49] *In re Schrader* 22 F.3d 290, 30 USPQ2d 1455.

this contradiction in the system is now missing. The highly effective opposition to the CII Directive through the collaboration of the open source and SME groupings has essentially made it politically difficult to move forward. The opponents to software patents have suggested that this is a welcome position to have arrived at: that Europe has been saved from the dangers of software patents. This does not appear to me to be true: business method patents have become easier to gain than traditional (more deserving, it might be thought) software inventions, and even these latter can gain some measure of protection through being re-constructed as device. The result of the CII Directive debate has perhaps been a victory of form over substance for the opponents of software patents.

The success of the software opponents has been a matter of form rather than substance because it has not stopped any patenting of software applications which would in any event have moved to grant. It may even have reinforced the view that the safest way for patent offices to continue is with the well-tried technical contribution approach, despite the conceptual confusion which surrounds it. In the view of this author this is undesirable, first because there can be no advantage in having a patent system which encourages conceptual confusion, and, second, because it enables a more problematic entity (business methods) to be more easily protected than it does a properly technical field such as computer science.

Clearly, however, the opponents do have a strong case in objecting to protection, but the case is really one which is best made against the workings of the patent system as a whole, particularly in the way in which it affects the small enterprise. All agree that SMEs find problems with the system,[50] not just with software. It may therefore take a more radical approach to the system as a whole – or the environment in which SMEs operate – before opponents feel at all happy about even the currently available protections. There will always be opposition to the introduction of intellectual property rights: it is only those with poor intellectual eyesight who see that IPRs are always only of benefit and never of detriment. But those who are seeking a world where 'small is beautiful' are, I think, destined to fail.

What of the future? We look at this in the final chapter and assess whether it will be possible to build a patent system which views the 'mere data processing person' as someone capable of producing inventions on his own.

[50] And not just SMEs – D. Crouch, in his Patently-O Blog (http://www.patentlyo.com/patent/2006/12/women_as_patent.html) notes that women are not being encouraged to innovate through the patent system either. A lively debate issued on the Comment facility of Crouch's blog. Women are well represented in computer science: is it conceivable that software patenting would increase the proportion of patents awarded to women inventors?

6 The third way: between patent and copyright?

The patent system is essentially weak and vulnerable even in the more industrialised of modern societies, but it does confer some advantages, as we have shown, and it is an important protection for the small firm and the small man. On balance it is a valuable institution, but its economic value overall is quite modest, and it is desirable that extravagant claims should not be made on its behalf.[1]

Introduction

The thrust of this text has been a mild defence of software patenting, primarily suggesting that:

- A new technology should be viewed as a technology on its own merits rather than via the legal fiction that it is something else.
- If invention in technology is to be protected as a social good, then why should invention in software technology not be so similarly protected? It is surely as difficult to invent in the programmed sphere as in any other.
- Arising from a form of 'technological determinism',[2] even if we are against software protection, there is – in the longer term – little which can be done to prevent it: since even if one does not agree with the pro-patenting perspective the fact that the malleability of software and its ability to undertake a 'machine-like' transformation in parallel to its taking over the role of the machine task (that is, from analogue control to digital control) quickly meant that it was near impossible (in terms of overall coherence of the system) to prevent patent applications from dressing themselves in the required format and gaining protection.

The mildness of this defence, however, is due to the inherent weakness of the patent model itself. In some fields such as pharmaceuticals it is

[1] C. T. Taylor and Z. A. Silberston, *The Economic Impact of the Patent System: A Study of the British Experience* (Cambridge: Cambridge University Press, 1973), p. 365.

[2] It is not, of course, actually technological determinism because so much of the force being applied is the social and economic pressures from vested interests.

probably essential, but in the software field it is not entirely proven that social benefit requires an expensive system of monopoly to defend what may well be a very minor technical advantage or that the system operates in any more rational manner than that of a lottery. Neither is it clear that a well-formulated set of software claims is particularly difficult to work around – unless the examiner has allowed patenting of the problem rather than one solution. Many analysts of the system have evidenced a similarly near-neutral feeling towards it – Taylor and Silberston's quotation at the head of this chapter summing up a general sentiment that it is useful, but by no means as useful as its most zealous proponents suggest.

Given, then, the lack of clear evidence that the patent regime is the most perfect system for achieving the desired goals with respect to the protection of software, is there another strategy which might be used to target these goals and which keeps both the open source community and many commercial developers happy? In the unlikely event that we were given a blank canvas and support from the various interests involved in the patent system and in software production, is there a 'third way' between patent and copyright, perhaps, which might produce the perfect system? Or, more realistically, at least produce a better and more appropriate system for software? Even if we were the strongest defender of the patent system, it is difficult to suggest that the system could not be improved in order to cope with the needs of this new technological artefact. For example, we have already noted that there are problems in the patent protection of software – the nature of what it is in software that we are protecting, for example, remains unclear: is it Moor's 'theory', 'model' or 'code' or all three of them which protection is to cover?[3]

Copyright protection of software, too, could certainly be clarified to cope with its new field of application. It has been the mainstay of program protection for two or three decades, and has, through TRIPS, become the internationally agreed form of protection for software; but there are many reasons why we might not protect *all* elements of a program through the copyright system – such as the core 'look and feel' of programs – since the regime has such a low level of creativity which has to be overcome and too much copyright protection of the non-literal may be over-protection. Indeed, in the UK which is out of step with many other European jurisdictions there is no requirement for creativity at all, simply originality. However, this does not mean that a program does not need copyright protection at all: the costs of copying others' digital versions *vis-à-vis* the production and development costs of manufacturing original products

[3] J. H. Moor, 'Three Myths of Computer Science', *British Journal for the Philosophy of Science* 29(3) (1978), 213–22.

are so divergent that few would argue that protection from literal copying is not required. Many non-digital products are protected simply by the fact that expensive manufacturing plant is required, and this plant's existence will be known to rights holders – historically, for example, the automotive industry would only seek patent protection in those countries where there was an automotive industry[4] rather than in all those countries where the produce is sold. This is certainly not the case with digital products, where mass copying requires only the most basic of equipment and resources. With these products, the right to prohibit direct *copying* through copyright provides a strong legal, if not necessarily enforceable,[5] protection.

Though few would argue that clear, literal, commercial copying of software should not be an infringing act, copyright offers more subtle and complex protections than just literal protection. Yet the limits of these remain unclear when we move away from literary works. For example, whereas the expression is the main focus of protection, the closer that the idea becomes to that expression then the more confused becomes what is the protectable subject matter. As Hand pointed out,[6] there is no mechanism which allows a clear division between idea and expression and the location of each on continuum must be found through analysis of the fact situation. This task is no easier when it comes to software. In Chapter 2, for example, we saw that there are conceptual problems related to the notion of idea and expression which arise when copyright is used to protect software and the courts themselves have been unclear about what is expression and what is idea.[7]

[4] The move of manufacturing across to the Far East is changing this situation: counterfeit goods can be produced outwith the direct view of the rights owners. See the International AntiCounterfeiting Coalition at http://www.iacc.org for the anti-piracy view. Organisations such as this have been quick to rework their rhetoric and to point to links between terrorists/organised crime and theft of their property rights: 'On February 28, 2003, Mohamad Hammoud was sentenced to 155 years in prison for helping to lead a cigarette smuggling operation that sent money to Hezbollah.' See http://www.iacc.org/resources/facts_on_fakes.pdf. The goal is to pass the costs of enforcement over to the state by emphasising the criminal/terrorist act rather than the civil infringement. Of related interest is *R* v. *Johnstone* [2003] UKHL 28 and the use of trademarks in bootlegging operations.

[5] My first question to intellectual property students is always to ask whether they have a right in the UK to make backups of their music collection or transfer this to other formats such as MP3 – many still errantly believe they do. This was raised in the *Gowers Review of Intellectual Property*, suggesting that rights which can't seemingly be enforced should be changed (http://www.hm-treasury.gov.uk/media/583/91/pbr06_gowers_report_755.pdf).

[6] 'Upon any work . . . a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out.' *Nichols* v. *Universal Pictures Corp.*, 45 F.2d (2d Cir. 1930) at 121.

[7] A good historical overview of the development and changing view of idea/expression in US law is found in E. Samuels, 'The Idea-Expression Dichotomy In Copyright Law', *Tenn. L. Rev.* 56 (1989), 321. His argument is that software caused a change from a bias towards

The lack of clarity of where the unprotectable idea lies in software is one perceived problem with copyright in software. Another is that copyright is a form of protection which many would argue is best suited for literary rather than functional artefacts – as many critics of copyright software protection have pointed out, few people buy software in order to read it or to look upon its aesthetic qualities and, even when it is made commercially available, it is usually in the form of object code which makes it much less readable and even less aesthetically pleasing. Those who do purchase software, do so for *what it does*: that is, its functionality, the protection of which has never been a core aim of copyright. There are other problems which we did not investigate in Chapter 2 which also arise from the copyright system: the 'spare parts' problem where one may wish to make similar kinds of alterations or additions to a program that one might make to a purchased machine, but cannot due to copyright and reverse engineering prohibitions.[8] In effect, copyright protection has been formally given to computer programs, but the nature of software has meant that, in important ways, there are differences in protection between a literary or artistic work and software – primarily where there is a conflict with the functionality of software as in, for example, the exemption of programming languages from copyright protection.[9]

The conclusion which we might draw – indeed, that critics have drawn – is that software is not best suited to either patent or copyright protections.

presumption of there being an 'idea' towards presumption of 'expression' and that: 'A survey of the history of the idea-expression dichotomy and a comparison with the application and policies of related doctrines suggest that the idea-expression dichotomy simply does not work very well as a categorical test of copyrightability.' An earlier critical perspective is that of S. Beyer, 'The Uneasy Case for Copyright: a study of copyright in books, photocopies and computer programs', *Harvard Law Review* 84 (1971), 291. An even earlier investigation of the nature of protection for 'literary' work is S. Lichtenstein, 'Study of the term "Writings" in the Copyright Clause of the Constitution', *N.Y.U.L.R.* 31 (1956), 1263.

[8] There is no common European approach to this (though it is currently more design than copyright oriented) but we can see the kinds of problems from the US in the interesting *Lexmark International, Inc.* v. *Static Control Components, Inc.* Civ. Act. No. 02-571-KSF (E. D. Kentucky, 27 February 2003) where Lexmark tried to ensure users were able to use only the company's own toner cartridges by including a program on a chip on the cartridge. Only Lexmark cartridges would thus work with the printer. Judgement was originally for Lexmark arrived at under protection by 'technological measure' that 'controls access' to a copyrighted work. This was overruled in *Lexmark International, Inc.* v. *Static Control Components, Inc.* 387 F.3d 522, 72 U.S.P.Q.2d (BNA) 1839 (6th Cir. 2004). Europe is currently considering a proposal for a Directive Amending Directive 98/71/EC on the Legal Protection of Designs, COM(2004) 582 final, 2004/0203 (COD). Although directed at the automotive industry, the language appears to also potentially cover software 'design'.

[9] Discussed in Chapter 2.

Given these views of the limitations of patent and copyright protection for software, proponents have thus argued that we need 'new thinking' – a form of protection which might give one or more of:

(a) the protection of program functionality which is currently offered by the patent system but in a more software-suitable form;
(b) literal protection from the copyright regime;
(c) protection of new developments which may not fit notions of 'invention' but which are new and useful.

Such a form of new protection is difficult to imagine and there have, in fact, been very few proposals which have been taken at all seriously, although there has been a variety of proposals which entail some tinkering with the patent system as it stands. For example, Cohen suggested that since – in his view – software is twofold in type (operating system and applications program), then due to the special network effect of the former, it should not receive full patent protection, but instead a *sui generis* form should be provided which is 'limited only to those software platform components which create compatibility (e.g. APIs)'.[10]

In this chapter, though, there is no attempt at a taxonomy of proposals and we will look at three of the many prominent schemes, and consider whether they do offer the claimed advantages over the present patent system, with its warts and all.

The 'Manifesto'

The patent system is based upon the supposition that technical advance is the most important element in the marketplace and that such advance will be rewarded with commercial advantage of some sort – for example, licence income or 'space' in which to operate. It is anecdotally clear that this is not always the case: there have been many advances which have not been successful for a variety of reasons, such as reluctance to change (as Grace Hopper pointed out[11]). It is clear that, in the world of software, one important reason for non-success in the computer area is because of network effects, yet the patent system pays little heed to this important element of the marketplace, concentrating instead upon the notion that

[10] S. A. Cohen, 'To Innovate or Not to Innovate, That is the Question: The Functions, Failures, and Foibles of the Reward Function Theory of Patent Law in Relation to Computer Software Platforms', *Mich. Telecomm. Tech. L. Rev.* 5 (1999), 1. As with other similar approaches, the rationale for the amendments comes from the asserted failure of the patent system to properly fulfill its economic 'reward' functions.

[11] See Chapter 1: 'It was a selling job to get people to try it. I think with any new idea, because people are allergic to change, you have to get out and sell the idea.'

technical advance and invention are primary. In fact, the patent system pays almost no attention to the marketplace at all.[12]

One proposal – *A Manifesto concerning the legal protection of computer programs*[13] – attempted to reverse this traditional perspective and look at the protection of software with a much more market-oriented approach.

The Manifesto was published in 1994 and although it is historical – in terms of protection having moved on in the past decade or so – its authors still appear to support this as a potentially substantive advance in useful protection.[14] At that date patent protection for software was available to those who knew how to compose the required claims, but much more important was the then recent US copyright litigation, which had focused on the 'look and feel' of programs where innovative developers attempted to protect their investment from those who copied without incurring the research and development costs – that is, producers in the 'clone' market. The Manifesto is thus primarily a response to the failure – as perceived – of the copyright system, but also critically appraises the patent system as failing the marketplace, too. For example, the authors suggest that '[n]o-one should have even a short-term monopoly for being the first to think of "computerizing" certain functions' – a recipe which would certainly undermine any possibility of patenting business methods (the patent example they cite is US5,105,184, which is clearly a business method based on advertising techniques rather than traditional computer science – 'Methods for displaying and integrating commercial advertisements with computer software'[15]). More generally, they take the view that patents

[12] In Europe commercial success is dealt with more critically than in the US: 'In T 478/91 too, commercial success was not regarded as indicative of inventive step. The board pointed out that it was well known that the commercial success of a product could just as easily be due to factors other than its properties, in particular more streamlined manufacture, a market monopoly, advertising campaigns or efficient selling technique (see T 270/84, T 257/91, T 712/92)', *Case Law of the Boards of Appeal of the European Patent Office*, 4th edn, Legal Research Service for the Boards of Appeal, December 2001, p. 137. See also the more recent T1212/01.

[13] P. Samuelson, R. Davis, M. D. Kapor and J. H. Reichman, 'A Manifesto concerning the legal protection of computer programs', *Columbia L. Rev.* 94 (1994), 2308. The approach has been called 'Radical Revisionism' – see J. B. Hawkins, Book Review, *Harvard J. of Law and Technology* 9(1) (1996).

[14] See, e.g., P. Samuelson, 'Mapping The Digital Public Domain: Threats And Opportunities', *Law & Contemp. Probs.* 66 (2003), 147: 'In addition, I have endorsed a short term of anti-cloning protection for industrial compilations of applied industrial know-how. Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs* . . . '

[15] This patent offers a highly market-oriented approach and appears – to this author – to fit well with the Manifesto proposal – perhaps innovative but not particularly inventive. The patent was granted 1992, and contains one claim: 'A small to a full screen commercial advertisement is to be integrated with different types of screens generated by a computer

aim for the wrong target – that they protect 'methods for achieving results' rather than the results themselves. Is this the first step towards a radical, revisionist approach? Methods of manufacture are intrinsic to patents and a criticism and replacement of a methods-based approach would involve a substantial step away from patent protection.

The Manifesto argues that programs are different from other technologies and that any protection should take that difference into account. The Manifesto takes a computer science-oriented approach, arguing that it is not so much the text of the program which is important but that the program is a type of virtual machine which is organised upon a variety of conceptual metaphors, much as outlined in earlier chapters in this text. Along with the critics from the open source movement, the Manifesto argues that programs are incremental and cumulative in character – that is, that often the most important aspects of a program are below the level required for the inventive height hurdle of the patent system. Unlike the open source community which prefers no protection for such improvements, the Manifesto suggests that this is a form of under-protection and requires to be remedied. The Manifesto argues that the litigation which was seen over 'look and feel' where developers attempted to protect their investment was a sign of a deeper problem – that the existing protections available did not protect what was really valuable in software.

The value of software, the Manifesto authors claim, is in its behaviour rather than the text: '[b]ehaviour is not the only source of value in a program, but it is the most important ... when buying software in the retail market, consumers buy behaviour and nothing more.'[16] Since it is the behaviour which is valuable, then the Manifesto suggests that behaviour is where protection should be focused.

Many commentators would agree with the Manifesto that it is certainly the behaviour of programs which is important. However, focusing on this, the Manifesto implies, requires a new way of protecting software. In effect, what this new protection – which is to replace copyright and patent – would protect, is a program from being *cloned*. Cloning a program means, to the Manifesto authors, taking the valuable element of a program without being required to undertake the effort and design involved in successful software development. It is, in this view, 'free piggybacking' and

software; and such commercial advertisement is to be integrated by modifying and/or inserting in a data entry, user interface, menu, prompt, help, edit, report, maintenance, error, action, game, sign on and off, and/or other similar screens in such software, wherein said small to a full screen commercial advertisement is to be placed in the different parts of a computer software so that such commercial advertisement becomes an integral part of such software, and yet does not interfere with or alter the original function of such software, and is not necessary for the computer software to function.'

[16] P. 2318.

thus undesirable, since unlike the more traditional marketing technique of 'piggybacking',[17] it is done without the agreement of the partner. Cloning software is problematic because the innovations in software which make it valuable in the marketplace are, to the Manifesto, highly vulnerable because they 'lie on the surface' of the product and are easy to view. In effect, it is because software differs from other products by making the acquisition of the value in the program easy that cloning protection becomes necessary:

> Because software bears the bulk of its know-how on or near the surface of the product, it is more vulnerable than traditional industrial products have been to trivial acquisitions of behavioural equivalence.[18]

Cloning, in industrial products, is hardly controversial: reverse engineering a rival product to utilise elements which are not protected is the day-to-day business of a research and development department.[19] Trade secret law has developed as a means to regulate what is permissible behaviour – that is, if the information can be seen then it is not protectable.[20] Yet, in the Manifesto, it becomes an activity which is seen as highly problematic. This view arises from the perception that it is too easy to extract the value from a software product and therefore it is the source of market-destructive effects: if the cost of innovation is substantial enough, then it cannot (as a tactic) survive the onslaught from copying of the valuable behaviour of the program. But the Manifesto suggests that cloning is not, however, something which should be totally halted, since there can be occasions when it benefits the market to have such copying. Rather, '[i]nstead of trying to stop dependent creation in software, the law should regulate how rapidly certain kinds of dependently created products can be introduced to the market and under what terms.'[21]

[17] 'Whether you are a mature company trying to break into a new market or a new company just trying to break ground, co-marketing can help you capture market share on a shoestring. By piggybacking on the strength and positive associations of a partner's brand, startups can get their first, much-needed shot of validation. Companies that are more established can leverage the brand and market reach of a partner to help them very quickly pick up the credibility and awareness needed to play in new market segments.' *Businessleader.com* 14(1) (2002).

[18] P. 2337.

[19] Henry Ford wrote in 1922: 'We study every car in order to discover if it has features that might be developed and adapted. If any one has anything better than we have we want to know it, and for that reason we buy one of every new car that comes out. Usually the car is used for a while, put through a road text, taken apart, and studied as to how and of what everything is made.' H. Ford, *My Life and Work* (Whitefish, MT: Kessinger Publishing, 2003), pp 145–6.

[20] Thus, broadly, so long as there has been no breach of confidence from someone with know-how, reverse engineering is allowable.

[21] P. 2380.

Thus the new protection focuses on extent of cloning and length of protection from cloning, suggesting that a legal regime should be constructed which essentially gives developers a protected 'lead time' in which they can take innovative products to market without fear of their innovations appearing in rival software.

It is at this point that things become more confused, since, as the authors suggest, there is a wide variety of cloning behaviours which range from a direct duplication of code, through exact replicas of behaviour, partial replicas, and even such elements as add-on programs, where developers produce interacting programs which feed off the behaviour of the original. That this latter should be deemed potentially contributing to market failure is because 'one can argue, from a market-oriented standpoint, that the developer of an add-on-program builds on the underlying developer's research and development and should not be entirely exempt from contributions to these costs'.[22] A critic might suggest that it will be as difficult to determine cloning as it is to determine inventive height or where on the idea/expression continuum this lies, but the authors think that: '[a]n anti-cloning standard would also be more predictable than a substantial similarity standard. Innovators, imitators, and courts can all be expected to recognize near-clones without much difficulty.'[23]

An underlying idea of the Manifesto is the work of Reichman (one of the authors), who has argued that what is important about programs is that they are 'know-how' rather than product or textual product.[24] Such an assumption implies that what is really important to the program developer would be best protected through some form of trade secret protection, but the argument is that this know-how is difficult to protect in programs, since their behaviour is surface oriented. Given that trade secret protection is thus not appropriately available, Reichman argued that 'to protect interested parties against the misappropriation of unpatentable know-how has thus become a crucial issue for world economic development'.[25] He also criticised 'patent law's total pre-occupation with the "inventive step" . . .'[26] The language of the Manifesto – as in large part derived from Reichman's perspective – demonstrates a claim not to be an attempt to improve the patent system in order to handle software correctly, but to be a total denial of the utility of the patent system as a way to protect what is really valuable in software.

[22] P. 2405. [23] P. 2400.

[24] See, e.g., J. H. Reichman, 'Computer Programs As Applied Scientific Know-How: Implications of copyright protection for commercialized university research', *Vanderbilt Law Review* 42(3) (1989), 639–723. He is critical of the commercialisation of university work.

[25] P. 662. [26] P. 651.

The Manifesto suggests a shorter period of protection than that offered by the patent system, but does not give a specific date. Beyond the term of protection, the idea becomes public domain and, since it has been registered, offers the competition a means to browse and utilise no-longer-protected innovations.

The Manifesto is a complex document but the above outline gives the flavour of the approach. Innovation is seen to differ from invention. And the 'look and feel', we might conclude it is being argued – being surface behaviour and thus accessible to the purchaser – is more valuable in a software product than a technique which offers, say, ease and cost efficiency in program maintenance[27] and which is not visible on the surface of the product.

If the broad approach outlined by the authors is accepted, how might this be transformed into a working system? They give three options:

1. The option of doing nothing is considered but is deemed that relying upon patent and copyright regimes leads to more cloning (which is viewed negatively) and, because of poor quality patents being granted, to the impeding of competition.
2. Anti-cloning protection would provide artificial lead time for software developers. This would be granted automatically and last only so long as it gives lead time but not does impede others' incremental development or lead to *de facto* standards.
3. A registration system with some form of licensing for software modelled on existing libraries of algorithms, so that potential users can browse through a shelf of program ideas which might then be licensed from the developer.

The Manifesto has been published and considered for some years now and while there are certainly aspects which are interesting, there may be flaws in the thinking which have become more obvious over the passing years. We look next at the assumptions and also at the possible transformation of the protection system, even at this late date.

Does the Manifesto offer improvement to the system?

There is no doubt that if the Manifesto was to be implemented then it would be taken up by software developers as an extra means of protection

[27] A technique of producing easy-to-maintain programs would be potentially worth very many billions of euros given that maintenance is the most expensive part of program life. It has been estimated that annual software maintenance cost in USA to be more than $70 billion. See, e.g., the review of literature on costs in J. Koskinen, 'Software Maintenance Costs' (2003), online at http://www.cs.jyu.fi/~koskinen/smcosts.htm.

for software – few are the forms of IP protection which have no users. But the Manifesto is not an addendum to protection, but rather a replacement and has to be judged on whether it offers significantly improved results from both policy and pragmatic perspectives. Note that while the origin of the proposed system arose from criticisms of copyright protection, it was put forward as a replacement to patentability and can therefore be judged according to patent criteria. In order to determine whether there is an improvement offered, it is necessary to investigate the primary assumptions which the Manifesto authors cite and claim makes software and patent incompatible.

Is software really different?

The authors put forward the idea that there is something new about the technology of software which makes it different from previous technologies. That is certainly true – the steam engine was different from horse-harnessing technology, and it surely involved new ways to think about technology. For example, it required a different system of support – foundry, machine tool industry, coal mining technology (to which it contributed), etc. – which meant that it became part of a larger industrial complex and a forerunner of the complexity of the industrialised software industry we find today. However, software goes beyond these differences and its own particular divergence is that – as we have discussed – it melds together the physical and the virtual. The Manifesto focus on the difference between the text of a program and the functionality simply confuses us: it mixes up the description/implementation and the functionality in the same way that, for example, a combination of drawing/rods, nuts and bolts etc. is different from the functionality of the combination. Of course it is – one is visual description and atomic makeup of the device, while the other is the device in operation. If we are aware of that, then we can view each as simply providing a different window upon the same artefact, rather than saying that because we have different windows through which we view it, there is no single artefact. The problem is in the view, not in the device.

In some ways, however, the authors are correct in seeing that there is a perceived difference between the text of a program and its behaviour. We can note that judicial reasoning has lacked this latter clarity of what the view is: they have been railroaded into a copyright perspective by that form of protection and – as would have been argued by Wittgenstein – their vocabulary has affected their understanding. But that does not mean that we need to follow this incorrect judicial visualisation and suggest that it is anything more than a legal fiction.

Dealing with software as behaviour, we can thus see that there is really little difference between, on one hand, software as product and, on the other, machine/compound as product. Both show their externalised behaviour for very little effort. And it is, in all these products, the behaviour which is of commercial value, not the atomic sub-parts.

What is so special about being cumulative and incremental?

The Manifesto author's view of software development is that it occurs as small-scale improvements rather than involving great inventive leaps. In fact, this ties in very closely with a commonly held view of the patent system: i.e. that it rewards developments which are incremental and cumulative. This is certainly a well-agreed view of the system by many neutral observers and many non-neutral ones, too. Perhaps the patent system and software are well matched?

Of course, the rhetoric of the patent system is that it rewards 'invention', but this depends upon the level of inventive step set by office and court. It may be that, if the level were raised, there would be a large number of developments in software which would lie beneath that hurdle, but this would require policy justification that protection should be offered to these low-level advances, which is certainly implied in the Manifesto but there is insufficient evidence for such a major replacement to the patent system. Some countries already have a collaborating ('second tier') element which might handle such advances (the petty patent system which we look at below), so it may not require the denial of patentability to handle the developments which the Manifesto authors see as being below the hurdle of patent protection.

Isn't everything protected by patent a behavioural form?

The focus on copyright and the lack of US judicial desire to protect 'look and feel' to the level which the authors suggest is necessary, leads to an implication that software with its 'behaviour' is different from other products which might be protected. This is difficult to follow: all chemical processes, products, ways of manufacturing etc. are protectable and they all are defined by behaviour. It is the 'idea' in the patent which is the object of protection and there is no reason why that idea cannot include behaviour. Given that technology is the *applied* use of knowledge, then surely (if correctly examined) all patent specifications evidence applied behaviour.

Is cloning really problematic? Add-ons?

The Manifesto authors were struck by an event when the graphical user interface which was developed at Palo Alto by Xerox was taken by Apple after a visit to the former research facility. This may be the origin of their view that software is amenable to easy theft of surface behaviour and that anti-cloning behaviour was not protectable and should have been. However, there is another reading of this event which casts a different light upon it. That is that Xerox were under anti-trust observation by the Federal Trade Commission and were required to open their patent folio to competitors with very low cost licenses. This is the argument which Gregory Aharonian has put, suggesting: 'It isn't surprising then that in the late 1970's, Xerox employees were law or unprepared in dealing with closely protecting Xerox intellectual property like the GUI interface. Why bother to protect the apparently unprotectable?'[28]

Anti-cloning is central to the Manifesto, but given its centrality there is really little justification as to why it is so wrong: where is the empirical evidence on actual lost lead time, or the period required to produce a rival product in the marketplace, etc.? The reader might almost conclude that there is a moral rather than an economic argument being made against cloning. Such a stance may be entirely justifiable – though it may be morally repugnant – but does not fully take into account how judicial development of anti-cloning rights might move protection in unwarranted ways. That is, that when we suggest new rights, we need also to think about possible negative developments which may accrue from these. The critical reader would note that legal protection of domain names has effectively expanded quite considerably the nature of trade mark rights – what effect might the legal right of software cloning have on other related rights such as machine development? If software is protected from sub-inventive patenting, then surely (the argument will run) other non-digital products should also be protected.

The short discussion of 'add-on' interacting programs, too, suggests a very much more extensive system of rights than allowable by patent: if a program developer is enabled to halt interaction of other programs with his own program, it gives a huge licensing opportunity to the winner of the first to gain the network effect. The Manifesto authors appear to be

[28] G. Aharonian, *Patnews*, 22 April 1996. The critique cites *PTO Official Gazette*, 4 November 1980 and its listing of some 6,000 patents which were to be offered. Aharonian is pro-patent but critical of poor prior art searching, offering a service in this field. See http://www.iplaw-quality.com/. See M. Hiltzik, *Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age* (London: Orion Business Books, 2000) Ch. 23, for a journalistic view of the visit.

setting up a system much like rampant capitalism, where the winner takes all, rather than a system which rationally applies reward in a balanced form.

Clarity of predictability of anti-cloning?

The Manifesto suggests that it should be easy: 'Innovators, imitators, and courts can all be expected to recognize near-clones without much difficulty.' As the study of law in practice has demonstrated, in the face of litigation – as the court door beckons – there develops less and less confidence in one's case. One might suggest that a similar falling off in confidence level will be found in recognising 'near-clones' as the court doors near, especially since it is not clear how that might be defined as a legal concept.

Registration – isn't this just the patent system again?

Finally, we are left with a feeling that the Manifesto is really not as radical as its title suggests. It does not offer any radicalised new version of protection at all – it has very much in common with the petty patent approaches which are looking to protect sub-inventions seen as commercially valuable but not inventive. The registration process which is suggested by the Manifesto appears to follow the petty patent model very well, as we see below.

If we approach the Manifesto not as a total denial of the patent system, but as an attempt to bring in an extra level of sub-patent protection for innovation, then we can see that it could make sense and – if linked to the policy reasons which underlie petty patents – has a ready-made justification, even if it is one which the authors did not utilise.

A Software Petite Patent Act

The second of the revisionist models is that of Mark Paley,[29] and this is perhaps a proposal which has met with more positive responses from the patent community and also, because of its emphasis upon protection for only commercial software, appears to offer some respite to the open source community. It is also a model which more explicitly connects into the already existing utility/petty patent type of protection which is reasonably well known in Europe.

[29] M. A. Paley, 'A Model Software Petite Patent Act', *Computer & High Tech. L.J.* 12 (1996), 301–19.

The presumption of the Paley model is, once again, that there is either overprotection or underproduction of software from the traditional copyright and patent regimes and there also exists confusion because software falls between these two legal protections which 'pull software apart to protect it'[30] and thus, for example, 'courts are split on how to protect the way a program interacts with a user'.[31] Paley echoes some of the early judgments which called out for Congress to devise a new form of protection, more appropriate for this new technology. Paley, however, is also concerned with the wider international perspective, arguing that although TRIPS is supposed to provide worldwide copyright protection, in fact, opposition to enforcing it in many countries has left software (and US software in particular) unprotected. A new form of protection – that proposed by Paley – would provide 'a better solution by providing lesser-developed countries ... a real incentive for software protection ...'[32]

The criticisms that Paley makes are the common ones; that, for example, algorithms were not well protected by the patent system, as indicated by the confused reasoning which eventually ended with Alapatt[33] and '[r]egular patents simply do not fit the basic nature of algorithms and software'. Mostly, he argues, the cause is due to trying to fit software concepts under the rubric of process or machine. And, if concepts could be better defined by statute, the term of protection is still too long for an artefact which 'far exceeds the useful life span of many software and many algorithms'.[34] On the litigational front, he also suggests that small firms find it hard to win patent cases against large infringers:[35]

> [s]oftware CEOs say it is easy to get a software patent, but hard to enforce it without a protracted, costly legal battle, and the U.S. Supreme Court has never sided with the patent holder. A small company with limited resources can be defeated long before the trial date by a large company that strategically makes litigation expensive and time-consuming through delaying discovery and motions. A small company unable to continue litigation may be forced to settle on unfavourable terms.[36]

It is thus the more traditional criticisms which underpin Paley's thinking. The Manifesto argued that the primary problem arises from trying to protect 'inventions' rather than behaviours and that the solution is in protecting what the market sees as valuable. This is not the view that Paley takes – rather, he appears to suggest that the patent system can be revised to handle software more properly, without throwing out the whole

[30] P. 305. [31] P. 305. [32] P. 312.
[33] *In re Alapatt*, 33 F.3d 1526, 1540–41, 1545 (Fed. Cir. 1994). [34] P. 327.
[35] This follows the policy criticisms dealt with earlier in Chapter 3. [36] P. 329.

system. His model is based upon the following elements (with section numbers referring to his proposed Act):[37]

Focus is on algorithm protection: §222(a) 'Whoever invents an algorithm may obtain a patent therefore.' Why such an approach? Basically because Paley argues that algorithms are essentially the same as those non-literal elements which are protected/non-protected by copyright and thus are the core element of software which should be the focus of petty software patent protection.

Protection through compulsory/blanket licensing: §236(b) 'Any person may obtain a compulsory license to commercially exercise the granted rights of the patent, unless such license would be contrary to the public interest.' The Petty Patents thus moves away from being a monopoly where the owner can stop the idea being worked: the idea is still protected under Paley's scheme, but as a property which others can access on payment rather than as a full monopoly. The USPTO would establish a 'reasonable royalty' system.

A reverse engineering right: §233(l) 'A software patent shall not affect a user's right to learn the internal design of a program by studying and experimenting with disassembled or decompiled portions of, or all of, the program code. Paley thus takes an opposing view to the Manifesto, given that the latter viewed too close investigation of others' work as being detrimental to the marketplace.

Exhaustion of rights on first sale: Paley objects to the use of licensing (rather than sale) to attempt to reduce the user's rights through exhaustion at first sale. Licensing conditions affecting resale are thus void.

'Use' to follow trade mark practice: To deny protection for software which is promised ('vapourware') but not in the marketplace, only software which is 'commercially developed' will be protectable (§233(c)). The target of Paley is the software developer who advertises software which is not yet available in order to undermine competitors' completed products.

Simple application filing (including listing): This application format follows the traditional patent method requiring clear and complete disclosure and claims containing technical features and a concise statement of what is being protected. Listings of code will be part of the publication and registration scheme.

No examination: §225(a) 'The Commissioner shall not examine the invention until challenge has been made.' This element follows on from having a simple application system, and thus cuts the cost of application substantially.

Infringement findings based on abstraction, filtration and comparison: Though he suggests that the model could be international, it is clearly based upon the judicial thinking of the US.

No infringement for non-commercial software: §233(k) provides a defence of 'Nonprofit, Noncommercial use' to the taking of any protected idea.

Some of these ideas are indeed European in style, where the idea of petty patents are well known and, as we see below, have already been proposed as ways to support the small software developer in Europe.

[37] Intended as amendment to US patent law in Title 35.

Criticisms of Paley's model

There are several elements of Paley's model which require a critical analysis in order to make a judgement about the value of this revised form of protection:

Algorithms The assumption is that algorithms are what require protection. This appears to be a lawyer's view of computer science when – as we have outlined earlier – software is really a virtual machine where algorithms are important but not the only element worth protecting. The patent system protects *ideas* and one may conclude that Paley's model, when it moves away from the protection of a technical idea, is offering a significant under-protection, just the criticism he makes of the patent system itself.

Compulsory licensing One of the advantages to commercial firms of the monopoly from a patent is that it allows them to create a space within which they can develop and manoeuvre. It is, to use the Manifesto authors' idea, a market-led approach. If innovators under the Paley scheme have no right to exclude but only a right to receive what may be nominal royalties, then those innovators may feel the protection is of little real value. There are many examples of firms who do not use the patent system for particular ideas because it makes too public their lines of development: these would be the same firms who would see little use in protection which advertised their wares but gave them no control over who else was allowed to implement them.

Vapourware There always appears in these kinds of 'model schemes' a moral judgement on behaviours and here Paley follows this practice by attempting to outlaw (or, at least, reduce) what appears to be a very normal practice. That is, enticing potential customers to wait for a 'better system'. Whether the practice would be undermined by denying protection only to systems at a state of commercial readiness appears unlikely, given the value in the marketplace of upsetting a competitor's marketing strategy.

Examination It is easy to understand why examination is seen as problematical and is omitted from the proposed Act, but no examination may be even more problematical. We look at this below.

Non-commercial This aspect of Paley's model is attractive to the open source community, promising that developments which are not for

sale would be outwith protection. Unfortunately, as we see from the successful development of open source, what begins as a non-commercial program may end up being integrated within a commercial system or sold as a package with documentation and support. Is this latter commercial or not? In the UK, the Patents Act 1977 (in s. 60(5)(a)) provides an exception to infringement if the act 'is done privately and for purposes which are not commercial', which is similar to Paley's suggestion. However, the courts have tended to view this in a limiting manner[38] where an act carried out in private and non-commercially must not then become part of a commercial act. This is essentially how the open source community operates – a private development of code which then is used by users in commercial situations – and would not be a non-infringing act. Paley's suggestion might therefore not be as useful to the open source community as it first appears.

Paley's model has much in common with a proposed European Directive, and given that this is a more formalised option, we will look at it now. Many of the criticisms of the directive, however, are applicable to Paley's proposed regime.

The proposed European Utility Model[39]

The Manifesto and Paley proposals have an underlying theme of desiring protection for the smaller enterprise – protection of the small developer against the likes of Microsoft, for example. We find that this same theme of protection through lower-level protection has been a recent point of interest for the European Commission, who have proposed a European Utility Model.[40] This protection scheme came about when the Commission published – in 1997 – a proposal for the harmonisation of utility model protection. Surprisingly, given the later attitude of the Parliament, the MEPs suggested strengthening and extending the protection which the Commission had originally proposed to cover software. As the explanatory memorandum to the revised proposal makes clear,

[38] See *Smith Kline & French Laboratories Ltd* v. *Evans Medical Ltd* [1989] FSR 513.

[39] This is based on P. Leith, 'Utility Models and SMEs' *Journal of Information, Law and Technology* 2 (2000).

[40] Recitals of the revised directive suggest: '(5) Whereas small and medium-sized firms play a strategic role in relation to innovation and rapid response to market requirements; (6) Whereas there is a need for placing at the disposal of firms, and in particular small and medium-sized firms and researchers, an instrument which is cheap, rapid and easy to evaluate and apply; whereas the fees should therefore be as reasonable as possible for small firms, individual inventors and universities; (7) Whereas utility model protection is better suited than patent protection to technical inventions involving a specific level of inventiveness.'

'Parliament did not question the Commission's approach and the main features of the utility model as described in the original proposal were retained'. Importantly, Parliament proposed extending the scope of the directive to cover software. This was accepted by the Commission and appeared in the revised proposal.[41] It is clear that the European Parliament were agreeable to SME-friendly arguments, since they suggested that fees should be reduced by 50 per cent for small and medium-sized firms, individual inventors and universities.[42]

There had been much discussion in the early 1990s concerning the value and construction of 'second tier protection', since it became clear that the Commission were interested in this and had been promising a Green Paper on the topic. The worry that proposals from Germany's Max-Planck Institut required a UK counter led to the CIPA's conference at Brocket Hall in 1994.[43] A common view in UK industry was that the value of patents could be undermined by the free-flow of unexamined utility models. The image of a 'minefield' was frequently cited as a potential result of having easy access to protection, where these protections were hidden from view until they exploded in litigation under the unsuspecting manufacturer. Such protection, rather than helping Europe to revitalise innovation and development, could act as a barrier to innovation, since firms would not wish to expend money and develop in fields where the protections were unclear.

Protection under the directive would be towards inventions which were 'industrial'[44] but, rather than the technical fields covered being identical to that of the EPC, there is an exclusion of protection for chemical substances or processes.[45] A brief outline of the proposal is:

- Protection would *not* be at the Community level, and utility models would be national.
- There would be no common procedures for processing utility models (e.g. setting up of opposition systems) and these will be left to national governments to consider.

[41] COM (1999) 309, 12 July 1999.

[42] Though it is not clear how this idea would have been policed. It would not be difficult for large enterprises to set up smaller, independent companies to control their utility model portfolios. Whether universities – already funded significantly by the public purse – should be further subsidised is a further question. This proposal was not accepted by the Commission.

[43] CIPA, *Second Tier Protection: Report and Proceedings of a Symposium, 6–8th July* (1994). The Banks Committee had earlier rejected petty patenting. See M. A. L. Banks (Chairman), *The British Patent System: Report of the Committee to Examine the Patent System and Patent Law* (London: The Stationery Office, 1970), Cmnd. 4407.

[44] Which includes agriculture – Art. 7(1).

[45] These industries were unhappy with the idea of utility models. See Recital 13.

- Protection would be for a maximum of ten years;
- *Computer programs would be protectable*.
- Not protectable would be: discoveries, scientific theories and mathematical methods; aesthetic creations; schemes, rules and methods for performing mental acts or doing business; presentations of information;[46] anything contrary to public policy or morality; biological materials; chemical or pharmaceutical substances or processes;[47] surgical or therapeutic treatment.[48]
- Inventions must not form part of the state of the art. Thus prior art would not be geographically limited.
- There would be no grace period.
- *Inventions will be considered to have an inventive step, if* 'it is not very obvious to a person skilled in the art'.[49]
- Search examination will only be carried out on request (or in the event of litigation) and there will be no formal examination of validity. Costs of search may fall upon the non-holder.

In some ways, the general approach being taken was closer to that which was being requested by UK proponents – particularly with regard to level of inventive step.[50] Essentially, this lowering of the hurdle for protection is one which mirrors the Manifesto and Paley approaches – that something below the level of an invention should be protectable. Are there real differences between these models and the European one with regard to what is protectable? It is difficult to tell, since it would only be in the procedural examination of the Manifesto and Paley schemes whereby one could determine just what was actually being protected by the courts – my reading is that there might indeed be quite a common feel to what is being protected by all three systems.

No-one in the debate about protection had originally imagined a nationally based system, this being unusual given that the Commission's single-market approach appeared to require a harmonised transnational approach. But the problems which the Commission has been having with cost of translation for the Community Patent may well have undermined any attempts to produce a Community-wide system. The directive

[46] These – Art. 3(1a)–(1d) – are excepted 'as such' by Art. 3(2).
[47] These are non-protectable by Art. 4.
[48] These are not considered of industrial application under Art. 7.
[49] The Max-Planck Institut suggested that an alternative requirement to that of obviousness would be that the invention offers a 'practical advantage'. The proposed directive has included this and two requirements now exist – that of showing advantage and being not very obvious.
[50] See a 1995 view of the rival proposals in M. Llewelyn, 'Proposals for the Introduction of a Community Utility Model System: a UK Perspective', *Web JCLI* 2 (1997).

obviously hoped to overcome potential problems for the internal market by imposing internal Community exhaustion of rights.[51]

For those involved in litigation there are strategic advantages which arise from this form of protection – being able to claim infringement at an early stage prior to patent grant, splitting off utility model applications from patent applications for quicker protection, etc. As both a defensive and offensive tool, therefore, there is much to recommend utility model protection. Outside litigation, there are other reasons for considering utility model protection. In Ireland, one advantage has been that tax relief has been available on research at an earlier stage and at a lower inventive level than that with the patent. And, in a commercial patent framework where patent licences are sometimes exchanged and traded by size of document pile rather than value of the claims, there is an advantage in having more pieces of paper rather than fewer.

Thus, despite the concentration of debate upon the usefulness of the utility model to the SME, it is unlikely that large businesses will not see advantage in using these devices for the same tactical reasons that smaller enterprises will. Indeed, there is no evidence that, in those countries where utility model protection has been available, SMEs have been their primary users. It therefore does not seem possible that the new directive will encourage a situation where SME usage will be enhanced and non-SME usage limited: we should expect larger enterprises to make more use of utility model protection in the new environment because these enterprises have more IP knowledge than the typical SME. It has been suggested that one reason for the limited use of the utility model protection in Germany, for example, by US and Japanese firms has been the cost of translation. Those non-European firms who have been filing European applications have been doing so in English, and have had no inexpensive route towards *Gebrauchsmuster* protection. If the directive is implemented, and the UK is required to provide protection, it will be a relatively simple matter for those firms who file via the EPO and PCT routes in English to prepare an English-language utility model at low cost and file this as a UK utility model. Transnational filing is usually carried out by larger firms, so a benefit will flow to them.

Another example[52] of possible usage of the utility model is that a company produced many more inventions than they would typically wish to patent (due to expense, for example) but were always concerned that if these ideas did not reach the public domain in some way, then they might be protected by the competition. One current method of ensuring

[51] Art. 21.
[52] Given to the author by a patent attorney for a major computer manufacturer.

that unwanted ideas become part of the prior art is to publish in the journal *Research Disclosure*.[53] For a modest fee, companies can put their ideas into the public domain, free from future protection. Thus utility-model protection may be as effective and cheap a method of stopping competitors from protecting ideas, but also provide extra protection to surround a patent portfolio. Such an option is particularly attractive to the larger company with a large body of patents – rather than to the SME.

It is not just that the advantages which accrue to the SME can be used by larger firms, but that to the SME utility models can be dangerous devices. In Germany the German Patent Office has suggested that, while the *Gebrauchsmuster* system is successful and has increased in application and grant number, the unexamined nature means that holders must exercise care in their use:

> In competition, claiming this right may trigger off claims for damages against the utility model owner by the alleged infringer. The utility model owner must be extremely careful, and is personally responsible for taking into account and appreciating the relevant state of the art as well as estimating inventiveness.[54]

The proposed directive (Art. 26) locates the legal framework for utility model, insofar as it is not dealt with in the directive, as being that of national patent legislation. This means that remedies for aggrieved parties who have been recipients of groundless threats of proceedings in patent matters will also be available for those receiving such threats of proceedings in utility model matters.[55] Since it is well understood that – in the absence of professional advice – SMEs have particular problems in reading claims and assessing the value of prior art, a simple system which might encourage filing and accusations of infringement against competitors by SMEs themselves, could produce negative effects upon business health.

That there are difficulties with utility models is clear: they are not solely SME-friendly, nor without danger. However, they are being granted in ever increasing numbers and their dangers may sometimes be overstated. They do offer a cheap and cheerful complement to full-term patenting which has strategic, cost and convenience factors. Given a situation

[53] *Research Disclosure* New York: Emsworth Design, Inc.

[54] DPA, *Annual Report* (1995), p. 32.

[55] Patents Act 1977, s. 70(1). 'Where a person (whether or not the proprietor of, or entitled to any right in, a patent) by circulars, advertisements or otherwise threatens another person with proceedings for any infringement of a patent, a person aggrieved by the threats (whether or not he is the person to whom the threats are made) may, subject to subsection (4) below, bring proceedings in the Court against the person making the threats, claiming any relief mentioned in subsection (3) below.'

where reasonably well-known technical fields are being protected, there is no reason to believe that anarchy will follow implementation of utility model protection in those countries who do not have it.

Are software utility models problematical?

The situation becomes different when we consider protection of software under the proposed directive. There has been no long-term development of clear principles of the law of patent protection for software, and there is – as we saw in Chapter 5 – the potential for considerable debate over the results of the Technical Board of Appeal decisions with respect to the line delineating what is and what is not protected. This means that we have an area of law which is undergoing substantial development and discussion and can certainly not be described as 'clear': yet, the directive on utility models suggested opening this area to usage by SMEs, and doing so without the support which would arise from proper examination of claims and prior art.

In essence, it appears to this author that the problems which have generally been highlighted as problematical in the granting of software patents will be more so in the granting of utility model protection. These are set out below.

Inventive step Setting levels of patent grant is one of the black arts of the field, and one which, no matter how objective the attempt (through, for example, tests of problem and solution), is always difficult to decide. The EPO has managed to impose upon European patenting a relatively harmonised level of inventive step. It has done so by force of application numbers as well as through significant training and internal harmonising procedures. No doubt, after a period of time, levels of inventive step in software will be harmonised and will operate within relatively narrow bounds. We are not, though, at that stage and there must be some worry that the lower levels of inventive step which might be expected through utility model protection must be unclear.

After all, what is the notion of 'not very obvious' to mean in software terms when it is not yet clear what is to be protected as 'not obvious'?

For those who do not have sufficient experience of the patent system (in particular SMEs), it may be that what will be seen as 'not very obvious' (and also worth protecting) are those parts of the program which the copyright system has failed to protect: for example, the 'look and feel' of programs. Certainly, a new program's look and feel can be 'not very obvious' and can have significant advantages – both to smaller firms who wish to protect their ideas, and to larger companies wishing to

develop a brand image and common interface. It seems likely that there will be a host of utility model applications (untested and unexamined) requesting protection which industry has shown it is interested in gaining, but which has not been provided through the copyright system. Such a volume of untested utility models must cause extra costs for SMEs – in terms of applying for protection which is worthless, and also considering the merit of competitors' claims.

No doubt this documentation will be built up, but it will require significant resources and also significant input from those involved in software patenting (in, for example, opposition procedures) before the documentation is sufficient and reliable. We will thus have to live with a period when 'dodgy' patents will be awarded due to limited prior art being available and being accessible. It seems a dangerous tactic to bring utility model protection into a context where there is so much confusion.

Evidence of infringement For all those considering using the new European software protection regime, there must be concern about gathering of evidence and ensuring that infringement is not occurring. In many fields the techniques are well developed – in engineering the machine can be stripped to its fundamentals and in chemistry the substance can be analysed. This is not the case with computer software, given that the 1991 Directive on the Legal Protection of Computer Programs provided software builders with a right to prevent decompilation except under specific circumstances and towards specific goals. The extra right is even more controversial when the patent system is taken into account: the infringer who utilises protected inventions within a piece of software may be enabled to hide that infringing use, but particularly when the invention is not related to the outward appearance or 'surface' functionality of the program. A litigant who suspects infringement may himself have to carry out copyright infringement in order to determine whether this has actually taken place or not. The route around this evidence-collecting problem is to go outside Europe and to decompile in countries which do not have this reverse engineering right. This is a solution which is particularly suitable for larger enterprises, and less suitable for SMEs.

Too much protection for competitors? The directive obviously grew from the then-current Commission and European Parliament philosophy which suggested that more protection is better. The pro-protection philosophy may well be correct, but there is little evidence that more protection must necessarily translate into more pronounced European innovation. The inception of the EPC has increased the European

protection available for US and Japanese industries by producing a patent granting system which enables easy grant throughout Europe. With around 50 per cent of patents being awarded to non-European sources, one might suggest that more and lower-level protection may simply retard European innovations, particularly in the SME field, and also in the field of software which is currently driven by US research and development.

It is not so clear that we should accept that a software patenting philosophy which is robust enough, or has developed a community of practitioners and examiners who have the ability to ensure that the balance between rights owner and non-owner, is already here. Given this, it may be a serious mistake to consider that our ideas of software invention are at such an advanced state that we should consider allowing utility model protection, and encouraging SMEs in software production to look for protection to this kind of device.

Chemistry has been excepted from those areas receiving utility model protection. There are substantial arguments that software should also be excepted, at least until the working of the system is carried out in the required review.[56]

Whatever the advantages[57] which might have accrued to the small developer, substantial opposition to the idea was found via questionnaire[58] (though, with a very low response rate) and reported in 2002. There appear to be no attempts to carry forward this idea and the Parliament may now take a different approach to the inclusion of software in such a proposed directive.

Conclusion: are these alternatives workable?

In this chapter, it has been suggested that there is a commonality to the approaches arguing for a reduced hurdle for protection which differs from that of the patent system – 'anti-cloning' to the Manifesto, 'petty patent' protection of algorithms for Paley, and 'not very obvious' to the European Commission. The view that these proposals take is that a lesser hurdle is appropriate for both software and for the smaller enterprise. The argument put here in contradiction to this view would be that software is really

[56] Art. 28.

[57] In their analysis of the Australian experience, Mortiz and Christie find the system works but is also used by other countries. No specific mention was made of software in this paper. See S. L. Moritz and A. F. Christie, 'Second-Tier Patent Systems: the Australian experience' *European Intellectual Property Review* 4 (2006), 230.

[58] *Summary report of replies to the questionnaire on the impact of the Community utility model with a view to updating the Green Paper on protection by the utility model in the internal market* (SEC(2001)1307).

not primarily a cottage industry any more – instead it is a major industry, both in its own right and as an adjunct to the traditional engineering industries where analogue control has been replaced by digital control.

The proposals all tend to imply that protection for SMEs would not be used by larger concerns, which appears to this author to be wishful thinking.

This is not to say that these systems would not work: they all appear to be perfectly reasonable forms of protection which could well be legislated and would work as well as any other form of IP right. However, only the EU proposal is being put forward as an extension of the current system, whereas the other two see themselves as replacements for the patent system. The problem with this latter view is that there is no clear method for extracting software inventions from the patent system – inventors would still want patent protection for their ideas and just by making another option available does not mean that patent attorneys would give up on seeking patent protection for their clients. The result will thus be a combination of the failures – as Paley and the Manifesto argue – of the current system and those of a new untested and untried system. It may be that fears of this permutation offering a less perfect system than is currently on offer will make legislators think twice about developing a new and special software-oriented form of protection. We may, as Aubry has suggested, be better with the system we have than a new system with untried faults:

> The system is as imperfect as other human systems. After 350 years, it has been refined and balanced, but as a result it can be criticised for its costs and delays. However, as has been said on more than one occasion, no-one has yet thought of a better system. None of the alternatives so far seems likely to attain those particular objectives which the patent system despite its short-comings achieves day by day and week by week. That is why it is so widely used by the world's industries.[59]

The system does appear to work – though just why and how is still not completely clear to any of the researchers who have studied it – and thus may, through a process of systemic sluggishness, remain the only option for software invention. It may be that attempts to improve the patent system by adding alternative protections simply makes the system more unwieldy, more fractured and generally less desirable.

[59] J. M. Aubry, 'A Justification of the Patent System', in J. Phillips, *Patents in Perspective* (Oxford: ESC Publishing, 1985), p. 9.

7 Conclusion: dealing with and harmonising 'radical' technologies

It is more difficult to reject a patent application than to grant a patent.[1]

Introduction

This book began as an investigation into whether it was possible to find a mechanism to limit protection for software 'as such' – that is, whether there was an alternative to the relatively woolly concept of 'technical effect'. The project involved immersion in patent applications, file histories, granted specifications, looking for an understanding of how it might be possible to apply that mechanism for setting the clear line across which the European patent system might not cross. This author's conclusion is that there is little real hope of drawing such a line in the sand. Primarily this is due to the fluidity of technical ideas in computing, but also – importantly – due to the very nature of the patent granting system, where the art of persuasion is one of the main skills of the patent attorney. It is almost as though, when we look at the software and the patent system, we see a path down which we are being moved: there may be objects in the path (the open source movement and other opponents, hazy notions of 'technical'), but there is a near-steam-roller effect of almost technological and economic determinism which ensures that we will eventually arrive at a position where this radical technology is protected. We are almost at that point, but a failure to accord sufficient importance to the way that this new technology is constructed has not helped, and has perhaps supported the fears of opponents. For example, emphasis upon technical effect has undermined examination of the technology as such and has allowed what appears, to this author, to be a very low order of invention to be protected. The 'technical effect' approach

[1] From an interview with Prof. Erich Hausser (then President of the German Patent Office). See P. Leith, *Harmonisation of Intellectual Property in Europe: A Case Study in Patent Procedure* (London: Sweet and Maxwell, 1998), p. 143.

relied upon there being a community of experts who could see what was technical and inventive: that is, relying upon a social community and its shared understanding. However, a computer science-oriented reader of some European patent specifications might often wonder where the experts were when these were examined.

In this concluding chapter, there are several strands which are worth highlighting. These do not comprise any exhaustive list – there are many parts of the patent system relating to software which would benefit from examination in a more theoretical light. For example, the role of bureaucratic systems in legal harmonisation is an important topic for theorists of law, and even feminists could look usefully at issues: programming has always had an issue with status *vis-à-vis* hardware and the early history of programming is one where it was perceived in a derogatory way as 'women's work'. Indeed, a number of the most innovative developments in computing have arisen from women inventors. But has that perceived lack of status impinged upon our later views of whether software should be protected when male-dominated notions of technology are viewed as superior? On the development of substantive law, there are relevant issues of the integration of monopoly-based approaches more directly into intellectual property law. Procedural matters, too, can be developed with respect to the patent system: how do we get a better and cheaper system of patent litigation in Europe?

Software has changed how the courts view national borders and this change may make one wonder whether we are moving towards a more international system. For example, we noted the Menashe patent (in Chapter 1) related to a gambling system which was linked to a communications system. Clearly, a communications system could mean that only part of the invention might reside in a country where there is patent protection – the part, most probably, which does not evidence any protectable element. William Hill, a bookmaker, had a system which allowed punters to bet via computers. The server was located in Curaçao in the Netherlands Antilles and, thus they claimed, could not infringe a European patent which had entered the UK national phase because the host computer was abroad. The judge was asked to decide whether a system so dispersed infringed under UK patent law or not. In a relatively short judgment, Jacob J took the view that what was important was the 'effect' of the patented system as a whole: 'No businessman would think for a moment that the effect of the invention is not within the UK when the whole point of the defendants' system is to get UK punters to play their system.' Looked at as a system, where the punters were based in the UK, clearly meant that the patent was being worked in the UK and thus – if the patent was valid – infringing:

If William Hill have a defence it must be that they are not using a system within the claims of the patent or that the patent is invalid. The wheeze of putting the host computer abroad is of no help to them.[2]

Likewise, in the BlackBerry litigation between RIM and NTP, the Court of Federal Appeals[3] also looked at the question of whether the location of servers abroad (in Canada in this instance) meant that the patent was not being infringed in the US. The court took the view that:

> we conclude that when two domestic users communicate via their BlackBerry devices, their use of the BlackBerry system occurs 'within the United States,' regardless of whether the messages exchanged between them may be transmitted outside of the United States at some point along their wireless journey.

The court followed on with reasoning similar to Jacob J that:

> Even though one of the accused components in RIM's BlackBerry system may not be physically located in the United States, it is beyond dispute that the location of the beneficial use and function of the whole operable system assembly is the United States.

This new technology of computer-based communications is clearly transnational and the national status of patents is becoming less important.

The patent granting system must be viewed in the context of litigation, a topic that has not been well researched in Europe. The European situation may not be quite as bad as in the US, where it has been held that:

> litigation has become an increasingly inefficient, ineffective and undesirable means of resolving patent related disputes ... unless the problems of cost and delay in patent litigation are addressed now, the central purpose of the patent system to provide an effective incentive for development and commercialization of new technology will be seriously eroded. Such an erosion could well prove a threat to the very existence of the patent system.[4]

but issues such as jurisdiction remain potential problems in a Europe where the 'Italian Torpedo' may or may not continue to exist.[5] Cost of either warranted or unwarranted litigation over infringement is certainly one of the

[2] *Menashe Business Mercantile Ltd and anor* v. *William Hill Organization Ltd* [2002] EWHC 397 (Pat), para. 25. The Court of Appeal affirmed the decision: *Menashe Business Mercantile Ltd and anor* v. *William Hill Organization Ltd* [2002] EWCA Civ 1702.

[3] *NTP, Inc.*, v. *Research In Motion, Ltd*, United States Court of Appeals for the Federal Circuit, 03–1615, 14 December 2004.

[4] Advisory Committee on Patent Law Reform, *A Report to the Secretary of Commerce* (Washington, DC: US Government Printing Office, 1992), p. 76, quoted in W. Kingston, *Enforcing Small Firms' Patent Rights* (University of Dublin Press 2000), NB-NA-17-032-EN-C.

[5] The 'Italian Torpedo' was a procedural manoeuvre to slow up litigation. Some commentators have suggested that the Italian Supreme Court has torpedoed this itself in the ruling in *BL Macchine* v. *Windmoeller & Holscher*, 6 November 2003. See a useful discussion from

fears of the European software SME groupings. There appears to be little comparative research conducted on the different procedural approaches in Germany, France, Italy and UK on litigation, but the rising level of hearings at the Munich courts suggests that patent holders view the German system as 'better', though it is not totally clear why. Germany has lower costs owing to having a much more restricted procedural system, but there may be other advantages, too.[6] Arbitration has been viewed in other legal areas as desirable as a means to reduce the costs of handling infringement issues, but whether it is an appropriate model when rights are awarded by the state requires more discussion.[7] And the finding of infringement by a software patent holder can be difficult even without a reverse engineering prohibition.[8]

These questions will not be dealt with here. Instead, there are several issues which are directly related to software protection which, it is suggested, require immediate attendance:

1. When we are dealing with a radical technology, we need a raised level of inventive step.
2. Where do we go now in terms of European harmonisation?
3. Finally, a very brief argument as to why patent protection for software may be of benefit to computer science as a developing discipline. The patent system is by no means perfect, and thus computer scientists have not been aggressive in demanding protection. But, to this author, it does seem that integration within the patent system could have benefit through the development of a more 'scientific' approach in computer 'science'.

Inventive step

The notion of 'inventive step' is one which is found in the EPC itself in Art. 56: 'An invention shall be considered as involving an inventive step if,

the EU-funded project (at http://www.ulb.ac.be/droit/ipit/), Judicial Cooperation In Matters Of Intellectual Property And Information Technology, *Cross-Border Litigation in Intellectual Property Matters in Europe Background Paper for the Heidelberg Workshop* (2006).

[6] See the Annual Reports, with figures of the BundesPatentGericht at http://www.bpatg.de. Also see K. Cremers, 'Determinants of Patent Litigation in Germany', Discussion Paper No. 04–72, Centre for European Economic Research (2004), available online at ftp://ftp.zew.de/pub/zew-docs/dp/dp0472.pdf

[7] Kingston believes there is evidence for its utility, but does leaves a number of issues aside in his discussion. See W. Kingston, 'The Case for Compulsory Arbitration – Empirical Evidence', *European Intellectual Property Review* 22(4) (2000), 154–8.

[8] K. Nichols, *Inventing Software: The Rise of 'computer-related' Patents* (Westport, CT: Quorum Books, 1998), pp 57–77 gives a useful example of this in his discussion of a text searching method which could be used for internet searching and which would be difficult to detect should it be utilised by a competitor.

having regard to the state of the art, it is not obvious to a person skilled in the art . . .' The conception of *step*, however, tends towards suggesting that we have an equal riser for all inventions and that this is somehow a given which is relatively easy for the examiner to follow. This is not the case: we would technically be better to talk about 'inventive height', since this emphasises that heights are as variable as lengths of string, whereas steps are usually not. When the EPO was instigated it came into a European patent world where there was quite a substantial difference between what each national office saw as the inventive step required to gain protection. For example, the German office was keen to promote a higher level of inventive step for several reasons – in part because the German office had historically had a higher requirement than other countries, but also because it saw itself as an aggressive competitor to the EPO, thus ensuring that the EPO's own standards of service to the patent community were kept up: competition was to be offered by speed of service to applicants, good search files and well-trained examiners. Importantly, one of the ways in which it saw itself as competition was in retaining a higher level of inventive step than the EPO. The then President of the German office suggested:

> And we have still kept our [higher] level of inventive step. It is more difficult to get a German national patent than a European patent and that is related to our level of inventive step. Connected to this high level is a broad scope of protection – with a higher level of inventive step you can give broader protection.
>
> There was a temptation for our examiners to be more generous when deciding levels, but we tried to keep our standards and many applicants now realise that this gives a better scope of protection. A lot of these applicants are coming back to our office.[9]

The aim was thus to have good examination, but to keep the level high so that the granted patent was viewed as being more likely to stand up to validity hearings. As Professor Hausser suggested, in the quotation at the beginning of this chapter, it would have been easier on his examiners to drop their level of inventive step, since granting patents was easier than finding reasons for not granting. Hausser was not a particular friend of the EPO, and his comments could be taken as a criticism of the EPO examination where, in his view, the EPO examiners might not be as good technically (though much better paid) and were prepared to give protection where it should not have been given. The EPO, of course, had a difficult task in their first few years. It had to determine a level which was appropriate and one which would not frighten away national applicants.

[9] From an interview with Prof. Erich Hausser (then President of the German Patent Office). See Leith, *Harmonisation*, p. 145.

It also had to take into account that the patents, when they moved into the national phase, would be considered by national courts each with a different attitude to inventive step. One of the earliest Board of Appeal decisions thus related to the setting of the height of the step where, in *Thermoplastic Sockets*, it was stated:

> In arriving at the above conclusion the Board has taken into consideration that patents granted under the EPC should have inventive step sufficient to ensure to the patentees a fair degree of certainty that if contested the validity of the patents will be upheld by national courts. This standard should therefore anyhow not be below what may be considered an average amongst the standards presently applied by the Contracting States.[10]

The *Leberl Study*, which was initiated by the EPO and looked into levels of inventive step, noted:

> The simple fact is that the level at which the inventive step requirement is pitched varies from one national office to another; but within each office, standards can vary considerably depending on the examiner handling the case.[11]

What might this mean for software examination? First, since inventive step is in large part a product of the problem and solution approach which utilises the closest prior art, we can see that it is possible to get the inventive step grossly wrong if there is an absence of prior art. Sometimes on reading granted patents (as with the document preparation examples noted in Chapter 4), one gets the feeling that, since little relevant prior art has been found or noticed, the invention is actually being given protection for its apparent novelty rather than anything more substantive. Second, it means that we cannot be sure that the level of inventive step being applied to computer-related inventions is similar to that of other fields, since there is no methodology which lets us measure and map each technical field. Third, it means that inventive step is a flexible notion rather than one which is concrete or fixed.

Many of the concerns of the opponents to software patenting arise because of what they perceive as obvious applications being given protection – that, for whatever reason, the inventive step is too low. While it is always difficult to analyse an application with respect to obviousness some years after it was published and after the field has developed technically, this author's own feeling is that there is more than a grain of truth in this criticism. It appears to be that, in a new technical area, the examiner's critical sense (based upon available prior art) is sometimes missing. Further, when workloads rise and the most important

[10] AECI/Thermoplastic Sockets, T1/81 (OJ 1981, 439).
[11] M. Vivian, 'Leberl Study', *Mitteilungen der Deutschen Patentanwälte* 84.Jg. (1993) 204.

measure of the success of an examiner is his throughput, it will always be possible for the examiner to take Professor Hausser's *dictum* to heart and grant a patent rather than reject an application, because it eases the workload.

This does not, though, seem to be an insuperable problem. If inventive step is a flexible notion, there is no reason why, in totally new fields such as computer program *as such*, the level of step cannot be raised. There will be objections from those attorneys whose main goal is protection at any cost,[12] but for confidence in the system as a whole, it appears that a higher inventive step in CII matters would be more useful than a lower one. Raising the level of inventive step will not solve all the problems in examining software as such, but it could be a major factor in undermining the opposition to granting patents on ideas which seem – to the computer science community – obvious.

Where do we go now?

If my prediction at the beginning of Chapter 1 is to become true – that is, that protection for software 'as such' is just around the corner, then how might it be achieved? There appear to be three paths: first, through a directive; second through some common court along the lines of that proposed for the Community Patent; and third through the EPO and its appeal structure. But which is most likely?

With reference to the first, it does not appear that the Commission, having had its nose bloodied over the CII Directive, will wish to enter the fight again.[13] It is possible, of course, that the environment will change and that the European Parliament will rethink its opposition to software patents if, for example, it realises what is actually being protected at present. However, there are no signs of this. The second path might have appeared possible until recently. The European Patent Litigation Agreement[14] appeared to be a method of moving forward by arranging a central patent court comprising a court of first instance and an appeal court:

[12] Leith, *Harmonisation*, vol.3, *Perspectives on Intellectual Property*, Chapter 5 deals with patent attorney perceptions of inventive step, etc.

[13] C. McCreevy, as European Commissioner for Internal Market and Services, 'The Commission's Work Programme for 2007', *European Parliament Committee on Legal Affairs* (2006): 'I will not bring a new initiative forward on this during my time as Commissioner for the Internal Market. I will leave this choice to my successor.'

[14] European Patent Organisation Working Party on Litigation, *Draft Agreement on the Establishment of a European Patent Litigation System* (2005), available online at http://www.european-patent-office.org/epo/epla/pdf/ewl0510.pdf.

Article 3 European Patent Judiciary
(1) A European Patent Judiciary is hereby set up to settle litigation concerning the infringement and validity of European patents effective in one or more of the Contracting States. The European Patent Judiciary shall have judicial, administrative and financial autonomy.
(2) The organs of the European Patent Judiciary shall be:
    (a) the European Patent Court, comprising the Court of First Instance, the Court of Appeal and a Registry;
    (b) the Administrative Committee.
(3) The European Patent Court shall perform the functions assigned to it by this Agreement.
(4) Subject to Article 5, the European Patent Court shall be supervised by the Administrative Committee.[15]

However, harmonisation of the judicial function continues to be problematic and although the idea of a European Patent Court has been given explicit support by many senior patents judges throughout Europe in what is known as the Second Venice Resolution,[16] it is not clear that the hurdles can be overcome as easily as some might hope. McCreevy, as European Commissioner for Internal Market and Services, was hardly enthusiastic about the likelihood of European involvement when he stated:

The European Patent Litigation Agreement is seen as a promising route towards more unitary jurisdiction. Therefore, I will ask my services to explore the possibilities of moving this project forward. However, you should be aware that there are some institutional hurdles to be tackled if the Community is to become involved in the EPLA initiative. Furthermore, stakeholders differ on the degree of centralisation or the nature of the local first instance courts.[17]

This lack of enthusiasm was to be well founded when the proposal was rejected by national governments in December 2006.[18]

[15] With the Administrative Committee appointing judges and deciding rules of procedure.
[16] http://www.eplaw.org/Downloads/Second%20Venice%20Resolution%20dated%204%20November%202006.pdf.
[17] 'Closing remarks at public hearing on future patent policy', *Public Discussion on Future Patent Policy in Europe*, Brussels, 12 July 2006.
[18] T. Buck, 'Hopes fade for EU patents reform initiative', *Financial Times*, 6 December 2006: '. . . the plan on Monday failed to secure backing from national governments, after several ministers called for the EU to pursue its own patent litigation regime. Mr McCreevy said the latest setback had made him "pessimistic" of making any meaningful progress on an issue seen by business leaders as crucial for the future of European companies. "Anything remotely concerning this patent area is fraught with minefields at every turn of the road," he said.'

The third option, that of involving the Boards of Appeal, is two-part. Practice could be amended by Board 3.5.1 or could be amended by the Enlarged Board of Appeal. The latter approach appears to be the suggestion of the English courts.[19] In *Aerotel* Jacob LJ suggested that the Enlarged Board might consider questions which the court had constructed:

(1) What is the correct approach to adopt in determining whether an invention relates to subject matter that is excluded under Article 52?
(2) How should those elements of a claim that relate to excluded subject matter be treated when assessing whether an invention is novel and inventive under Articles 54 and 56?
(3) And specifically:
    (a) Is an operative computer program loaded onto a medium such as a chip or hard drive of a computer excluded by Art. 52(2) unless it produces a technical effect, if so what is meant by 'technical effect'?
    (b) What are the key characteristics of the method of doing business exclusion?

This suggestion follows on from a general view in the UK that the Enlarged Board of Appeal is a relevant source of policy clarification, as was said by Neuberger LJ in *LG Philips* v. *Tatung*:[20] '[t]he Enlarged Board has developed the law on added matter in ways which can be said to involve superimposing a degree of policy over what had been perceived by the English courts as a relatively pure issue of principle.' Such a view may or may not be correct, but are these *Aerotel* questions ones which the Enlarged Board would be prepared to consider? Perhaps, but it seems unlikely that they will produce any solution different from that of the Boards of Appeal themselves – given that the Enlarged Board is made up of Boards of Appeal members with 'added externals' rather than being of different composition entirely.[21] Board 3.5.1 has had several opportunities to pass these questions on to the Enlarged Board but has not felt the need: these, in its view, are questions of a technical nature and do not

[19] 'It is formally no business of ours to define questions to be asked of an Enlarged Board of Appeal. What we say now is only put forward in case the President of the EPO finds it helpful. If he thinks it pointless or arrogant of us to go this far, he is of course entirely free to ignore all we say. Nonetheless in the hope that there is a spirit of co-operation between national courts and the EPO we ventured to ask the parties what questions might be posed by the President of an Enlarged Board pursuant to Art.112. As we have said the British Comptroller of Patents has encouraged us in this course.' Jacob LJ in *Aerotel Ltd* v. *Telco Holdings Ltd and others Rev 1* [2006] EWCA Civ 1371 at para. 75.

[20] *LG Philips LCD Co Ltd* v. *Tatung (UK) Ltd and others* [2006] EWCA Civ 1774.

[21] Details of current membership are contained in 'Notice concerning the composition of the Presidium of the Boards of Appeal for 2006', 19 December 2005, EPO.

require the legal input from the Enlarged Board. As Board 3.5.1 stated in their decision of 30 July 2003:[22]

> According to Article 112(1)(a) EPC, 'important points of law' . . . shall be referred by the Boards of Appeal to the Enlarged Board in order to ensure uniform application of the law. Questions which normally arise during proceedings before the Board and merely relate to the interpretation of the technical content of the patent application, the patent specification or the prior art documents, or are concerned with the assessment of novelty or inventive step, cannot normally be considered to warrant the referral of a question of law to the Enlarged Board of Appeal.

Which approach can be read as suggesting that the notion of 'technical effect' will be read as being concerned with the 'assessment of novelty or inventive step' and thus not appropriate for the Enlarged Board. The questions set by Jacob LJ thus potentially fall within what the Board of Appeal views as its own competence, rather than that of the Enlarged Board. Jacob, however, is suggesting that the Board itself is not required to make the referral, but that this should be done through the President of the EPO, which would be a possible route – under Art. 112 – but only if there is a difference of opinion between two Boards of Appeal.[23] This does not appear to be the case here: all the boards are happy with the concept of 'technical effect' and recent EPO overviews of the law relating to inventive step and technical contribution do not raise any particular problems of diverging interpretation.[24] Whether the Enlarged Board would in any event be happy to take on these questions is moot: their involvement to date in decisions has primarily been in the resolution of procedural questions rather than in the matter of substantive patent law.[25] As this author noted of G1/97 ('Request with a view to revision') in a study of the Boards of Appeal:

> The Enlarged Board of Appeal had thus the opportunity to deal with questions as to how it saw its role and its relationship with the Boards of Appeal, and also how the EPC related to the wider legal context of TRIPS. In its decision . . . the EBoA declined to see itself as a 'court of appeal', argued that it could not initiate such an extra appeal process and also suggested that TRIPS and the EPC were not incompatible with regard to review of decisions. This, too, was how the

[22] T 0367/01, 'Television set with improved remote control unit'.

[23] EPC Art. 112: '(b) the President of the European Patent Office may refer a point of law to the Enlarged Board of Appeal where two Boards of Appeal have given different decisions on that question.'

[24] See EPO, *Board of Appeal And Enlarged Board Of Appeal Case Law, Special edition OJ EPO* (2006).

[25] Their major input to substantive patent law has been G5/83 (Second medical indication), G 0002/88 (Friction reducing additive) and G1/98 (Transgenic plant/NOVARTIS II).

President of the EPO had seen the situation in his comments to the Enlarged Board – that the Board of Appeal decisions were final, that it was never the intention of the creators of the EPO to create 'its own procedural law' or 'creation of an entirely new judicial remedy'. The EBoA noted that not all countries who were signatories of the EPC had developed judicial review by statute, though it claimed that in 'the vast majority of cases' this was so.[26]

If the Enlarged Board had viewed their role as one of review of the Boards of Appeal (as, in fact, a 'court of appeal') which would allow a type of judicial review of decisions from the other boards, then it is difficult to see how this development could have been halted. It may even have been welcomed. The independence and standing of the Boards of Appeal would not necessarily have been undermined, and the general feeling amongst users of the EPO that procedural matters which impinged upon 'fair process' would be seriously dealt with would surely enhance the standing of the appeal process in the EPO. That appears to be the role which Jacob is seeking: unfortunately, it does not appear to be a role which the Enlarged Board itself desires, and this is most probably due to the structure and makeup of the Enlarged Board: it is simply not an independent review mechanism, but rather is a means for the various board members to agree a common approach amongst themselves.

This leaves any development within the sphere of the Board of Appeal itself, and most probably within the sphere of Board 3.5.1. Is there likely to be a change of tack away from the woolly concept of 'technical effect'? Perhaps – since Board 3.5.1 has been happy to discuss the possibility of TRIPS and its relationship to the EPC. For example, in T0276/99, the Board seemed to suggest that if there was a serious conflict between TRIPS, EU treaties and the EPC that it would be prepared to discuss this conflict:

15. The Boards of Appeal are bound by the provisions of the EPC (Article 23(3) EPC). What the appellant is seeking in his main and first auxiliary requests is against these provisions, and the Board cannot regard the appellant as having made out any serious case by reference to the TRIPS Agreement or the EU treaties that might justify allowing something forbidden by the EPC.[27]

T276/99 further suggested that: '. . . as the EPO Boards of Appeal are not a court or tribunal of an EU Member State, they do not have the status to refer a question to the Court of Justice of the European Union.' Is this an indication that they would feel confident about handling any relevant

[26] P. Leith, 'Judicial and Administrative Roles: The Patent Appellate System in a European Context', *Intellectual Property Quarterly* 1 (2001), 50–99.

[27] T 0276/99, 26 September 2001, 'Display device including a correction circuit, and correction circuit for use in said device'.

questions themselves rather than passing them on to another judicial body? Such a move would be politically controversial and would require a Chairman who was happy to put his head above the parapet, but, given the lack of logical coherence in the present situation, the present Chair may be prepared to undertake this integration of TRIPS within the EPC framework.

The important element of TRIPS, so far as patentability of software is concerned, is Art. 27, which refers to protection being available in 'all fields of technology':

> TRIPS Article 27: Patentable Subject Matter
>
> 1. Subject to the provisions of paragraphs 2 and 3, patents shall be available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application . . .

Arising from TRIPS is a developing view – and one which is being raised by parties in argument at the Boards of Appeal – that Art. 27 requires that software be protected as such since it is a 'field of technology'. TRIPS is potentially a highly important development in the protection of software, offering, as Cornish and Llewelyn suggest, 'a potency of novel order in the international relations of IPRs'.[28] If this author was a gambling man – a user of the system outlined in the Menashe patent, perhaps – he might suggest that a small bet might be laid on Board of Appeal 3.5.1 finally shutting off the software exclusion through integrating TRIPS into the interpretation of the EPC.

Conclusion: are patents of benefit to a radical technology?

The thrust of this book has been that software should be protected in its own right through the patent system. However, what has been proposed is a relatively limited view (i.e. 'traditional' programs) and it may be that we need to consider a broader view of where we go with the protection of innovation in the world of ICT. We are – after all – dealing with a radical technology which has now had some measure of protection from the patent system for almost a century and a half: Morse's US 1,647 of June 1840 included a claim to 'signs' which could be transmitted over distances.[29] The radical nature of this technology has been recognised by

[28] W. R. Cornish and D. Llewelyn, *Intellectual Property: Patents, Copyrights, Trademarks and Allied Rights*, 5th edn (London: Sweet & Maxwell, 2003).

[29] '3. The use, system, formation, and arrangement of type, and of signs, for transmitting intelligence between distant points by the application of electro-magnetism and metallic conductors combined with mechanism described in the foregoing specification.'

many cultural commentators such as Marshall McLuhan and Walter Ong, who posit that the most radical of technologies are always those which affect communication and distance (whether in space or time), and that the new 'electronic culture' could be expected to change society just as radically as had the previous communication revolutions of manuscript and print. There surely can be no-one who fails to see that this new means of communication is highly revolutionary: yet the patent system appears to deny this, by suggesting that we have to force the technology into an older device model. The situation of protecting this new technology through a device is akin to protecting chemistry through dressing up the inventions with test tubes and flasks and locating the invention in the laboratory hardware. Such an approach is hardly suitable for such a far-reaching technology.

The approach from computer science has not helped. Like most technologists, they have preferred to stay clear of lawyers, viewing them as more problem than solution. Thus the underlying assumption of the discipline has been similar to that of Garfinkel *et al.* and their belief that 'patents are bad for software'.[30] That is certainly one view, but it is not the only view. It could equally be the case that what the field of computing needs at present is a patent system to force upon it some measure of discipline. For example, we have already noted that there is a haphazard use of terminology in the field; there are also – to this author's eyes – frequent instances where the wheel is being reinvented over and over again; and peer review of claimed advances is almost-totally missing. A patent system which examined software on software's own terms may well be the mechanism which forces an improvement in the culture of computer science. The court and legal system will make claims to producing the 'truth' – certainly a conceit – but the courtroom does have the ability to force the participants to consider their assumptions. Technologies such as physics and chemistry have developed within a patent framework and it has done them – in the long term – no harm at all. They have means of clear communication, a developed sense of what constitutes a technical advance, and a robust attitude to testing claims of novelty. The patent system has not been entirely responsible for these developments, but it does not appear to have prevented a positive environment developing.

Computer science is not a field which has developed the kind of institutional and agreed frameworks for communication and peer review: far from it – it is a 'science' which is far from 'scientific'. Donald Knuth's *The*

[30] S. L. Garfinkel, R. M. Stallman and M. Kapor, 'Why Patents are Bad for Software', *Issues in Science and Technology* (Fall, 1991), 50–5.

Art of Computer Programming was an attempt to bring a methodological approach to computing, but it was one which was based in mathematical approaches. It is clear, almost 40 years later, that the mathematical approach has not brought rigour, or less bug-ridden programs, or helped to avert the 'software crisis' and the ability of the discipline to produce programs in complex environments on time, to cost, and to function as promised. This author's own view is that computing is closer to a more 'engineered approach', but not the kind of 'software engineering' which tries to use mathematical techniques. Rather, it is a complex combination of pure technology and social issue – and that social aspect is frequently to do with usability in its widest sense. If techniques can be found which will save many millions of euros from being wasted on catastrophic computer implementations,[31] then why should the inventors of these not be rewarded? If techniques can be found to take a computing artefact which is successful in the lab but not in practice out into the world, again why should the inventor not be rewarded?

The debate over protection for software 'as such' is sterile. Software is being protected in Europe. What is more important now is to ensure that it is being appropriately protected – particularly that over-protection is not available, and that examination is being carried out effectively. But it is also important that we detach ourselves from the current debate and attempt to locate just what it is in software and its application that we want to encourage – that is, how broad should our view of 'technology' be when we come to discuss the aims and goals of the patent system? Peter Prescott suggested in *CFPH LLC*, that '[w]e sense that we know "technology" when we see it',[32] which is certainly true, but highlights that we are all going around with different ideas in our heads about the meaning of the term. It appears to this author that now is the time to discuss more fully our disparate notions of technology and make explicit just what it is about software which we wish to protect and concentrate upon ensuring that the patent system does this openly and free from artifice.

[31] And few are the UK government projects which do not waste significant sums of money.
[32] *Patent Applications by CFPH LLC* [2005] EWHC 1589 (Pat).

Index

Cambridge Intellectual Property and Information Law

Titles in the series (formerly known as *Cambridge Studies in Intellectual Property Rights*)

Brad Sherman and Lionel Bently, *The Making of Modern Intellectual Property Law*
978 0 521 56363 5

Irini A. Stamatoudi, *Copyright and Multimedia Products: A Comparative Analysis*
978 0 521 80819 4

Pascal Kamina, *Film Copyright in the European Union*
978 0 521 77053 8

Huw Beverly-Smith, *The Commercial Appropriation of Personality*
978 0 521 80014 3

Mark J. Davison, *The Legal Protection of Databases*
978 0 521 80257 4

Robert Burrell and Allison Coleman, *Copyright Exceptions: The Digital Impact*
978 0 521 84726 1

Huw Beverly-Smith, Ansgar Ohly and Agnès Lucas-Schloetter, *Privacy, Property and Personality: Civil Law Perspectives on Commercial Appropriation*
978 0 521 82080 6

Philip Leith, *Software and Patents in Europe*
978 0 521 86839 6