Patenting programs as machines [1]

Philip Leith
Queen's University of Belfast

Abstract

It is clear that software is being protected in Europe. Not only is software being protected, but business methods are routinely receiving patent protection. Protection is being given when the applicant succeeds in recasting the software invention as hardware device or hardware device control. Thus, for example, in the recent *Aerotel* appeal judgment in the UK, a business method described in hardware terms was viewed as protectable but another business method (*Macrossan*) couched in software terms was not. [2]

The argument I wish to put forward is that the current approach – which was originally set out in *Vicom* [3] – has used a model of invention in computing which does not reflect how the software community views invention. Programs have been protected in the guise of 'devices' or 'machines' rather than as programs. This causes a mismatch between what should be protectable and what is protected and, to an extent, explains much of the opposition to software patents in Europe.

If software was to be examined and protected as software, would the opposition be resolved? Perhaps, but to get that position there remain specific problems to be overcome which concern the nature and examination of software: this requires a radical approach to cope with a radical technology before opponents might agree that the patent system serves the software marketplace.

Nymeyer & Vicom

[1] This article précis's a forthcoming (2007) text, *Software and Patents in Europe*, Cambridge University Press.

[2] *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." *Aerotel Ltd. v Telco Holdings Ltd & Ors Rev 1 [2006] EWCA Civ 1371 (27 October 2006)* Para 53. Emphasis added. At first instance Lewison J. had noted that, "[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 many general terms." *Aerotel Ltd. v Telco Holdings Ltd* [2006] EWHC 997 (Pat)

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

Prior to the inception of the European Patent Convention the patenting of software and also to what are now called business method patents was 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". [4] This broad-reaching 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.

As a useful example of this move we can consider the Nymeyer patent (GB1352742) which was filed in 1971 from an earlier US application [5], entitled "Improvements Relating to Data Processing" which in many ways was a precursor to the Signature patent [6] litigated in *State Street* [7]. The Nymeyer specification outlined that the invention related to the use of a computer to operate a market. Both Nymeyer and Signature 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 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." [8]

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:

> "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. …"

---

[4] 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".

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

[6] 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.

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

[8] *Application by IBM For The Revocation Of Letters Patent No. 1,352,742 In The Name Of Frederick Nymeyer*, Patents Appeal Tribunal , Monday, 16th 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* and is available online.

Such a description using terms like these 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 or Cobol or similar.

The court 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 [9] – but that 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 ..."

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 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.

UK practice briefly changed with the introduction of the European Patent Convention and the following of EPO practice. In 1984 Gert Kolle, a member of DG5, was writing that:

> "The chances of being granted a European *[software]* patent or a national *[software]* patent in a Continental European country are not good." [10]

Software related applications were thus routinely denied by examiners at the EPO until the first such appeal (*Vicom*) reached Board 3.5.1 who radically suggested it could be protectable due to the *technical* nature of a program on a machine:

---

[9] "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 …"
[10] Kolle G 1984 p. 61

> "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 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 [11] 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." [12]

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: [13]

> "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

---

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

[12] Van den Berg P., (1996) "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*, Carl Heymans Verlag., Munich. p. 33

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

process controlled by a suitable 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." [14]

Why did board of appeal 3.5.1 take this step? Because it was obvious to them that the software exclusion under the EPC was not practical. It was not practical because software – by the 1980s – was becoming a major part of all areas of technology. And, 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': they could *in toto* be viewed as machines. This is the solution of a practical 'community' rather than a legal concept; and this in part explains the difficulty courts have had with it.

What is wrong with *Vicom* ?

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 – the traditional model of 'machine'. That is, it attempted to distinguish between an abstract concept and technical signals ("technical contribution"). 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 the 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.

This machine-oriented approach causes difficulties in examination. For example, the Menashe patent (EP625760) 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 [15] 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.

[14] At p. 45.
[15] Examination report. 6 November 1997. Application 94303526.1

> The objective problem so stated clearly does have the required technical nature required by the EPC …" [16]

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.

It was not to be until T97/0935 (an IBM application [17]) that the Board 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." [18]

And suggested that a view that protection could not be given for the underlying program was, the board held, illogical. [19] The appeal was allowed, the application sent back to the examination division and the guidelines were amended to incorporate this new practice. [20] A wider conception of 'technical contribution' was thus developed but one which was still, to a very large extent, machine based (i.e. software controlling a machine was protectable).

The new approach certainly removed one illogical factor, while trying to keep other 'abstract' inventions outwith protection – particularly 'algorithms' and 'business methods'. Unfortunately, allowing protection for software *per se* simply opens up other illogical positions which the insightful patent attorney can attack to benefit his client.

Ignoring Software

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:

> "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

[16] Reply to examination report. 15 May 1998. Application 94303526.1
[17] This was granted as EP0457112. "Asynchronous resynchronization of a commit procedure".
[18] Para 9.3
[19] Para 9.8
[20] Guidelines, Part C, Chapter IV, 2. Inventions.

> partially understood. They are modelled to our permanent satisfaction rarely by our computer programs." [21]

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. It says that 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 most clearly in the use of diagrams by computer scientists when they explain what they are trying to do. 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 [22] : 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 maximizing power use the programmer – especially in the early days – was concerned with reducing memory usage and maximizing throughput to the processor. The thinking was the same, but the nature of the machine differed substantially – one tangible, one virtual.

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 most usually missing is that of the programmer. It is as though the programmer's view of technology was 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 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". [23] 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 *inform ation*, a procedure which involves no Reuleaux-like work performance except moving the contents of one data structure to another

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.

---

[21] Foreword to Abelson H., Sussman G.J. & Sussman J., (1985) *Structure and Interpretation of Computer Programs*, MIT Press.
[22] See Smiles S., (1861-62) *Lives of the engineers, with an account of their principal works: comprising also a history of inland communication in Britain*, J. Murray, London.
[23] Reuleaux F., (1876) *The Kinematics of Machinery*, p. 35. Online at historical.library.cornell.edu.

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 [24] which emphasises invention at the conceptual (that is virtual) level. And Hoare's view that 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". [25]

The unfortunate narrative of the EPO is that since the 1950s a radical 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 19th 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 uphold and thus we had *Vicom* and the later decisions. 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.

Examining Software

In software 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 performing 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

[24] Hansen, P. B. (2002) "The invention of concurrent programming", in Hansen P.B., (Ed.) *the Origin of Concurrent Programming: From Semaphores To Remote Procedure Calls*, Springer-Verlag, New York, New York, NY, pp. 3-61.
[25] C.A.R. Hoare, (1973) *Hints On Programming Language Design*, Stanford Artificial Intelligence Laboratory Memo AIM-224. STAN-CS-73-403.

commentators have suggested [26] – removed the need in software patents to have any substantive enabling information at all. One IBM attorney [27] 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 many abstract description provided.

But simply employing computer scientists and programmers to examine at the EPO – thus "listening to the programmer" – will not overcome all problems since software descriptions are different from those of, for example, chemical, engineering or electrical patents. They differ in ways such as;

- First, the underlying programmed code is usually ignored;
- Second, the programmer is working with virtual models which may be common (such as tables, stacks or trees) but which are in many areas likely to have been self-constructed by the programmer, or the programming team, or provided by other teams;
- Third, it is often the model from which the program is constructed which is at the heart of the 'inventive idea';
- Fourth, 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, say, diagrammatic representations of the model which differ according to the level of abstraction. 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 that 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.

The malleability of software is of huge benefit to the patent attorney – the invention can be relatively easily recast in any desired form to fit in with the formal needs of the examiner. This explains Beresford's encouragement to patent attorneys to think creatively about how they construct their software related applications:

---

[26] See for example Burk D.L. & Lemley M.A. (2003) "Policy Levers in Patent Law", *UC Berkeley Public Law Research Paper No. 135; Minnesota Public Law Research Paper No. 03-11*. "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"

[27] Flynn J.D., (2001) "Comments on the International Effort to Harmonize the Substantive Requirements of Patent Laws" IBM submission at *www.uspto.gov/web/offices/dcom/olia/harmonization/TAB42.pdf*.

> "[with] applications for computer systems for the performance of business methods, therefore, it is essential to give careful and *im aginative* 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." [28]

The examiners then find themselves in the unwelcome position of:

- having a technology described using the model of a different technology;
- having poor access to prior art (a commonly agreed problem);
- unable to discover whether the invention actually works; and
- not quite sure what he is examining – as noted by Moor [29] when he pointed to three ways to describe a software artefact: theory, model and code.

In this kind of environment where examiners are being urged to speed up examination and "[I]t is more difficult to reject a patent application than to grant a patent" [30] the suspicion must be that patents are being granted to inventions which are undeserving.

Where To Now?

The EC felt there are strong policy reasons for protecting software – e.g. 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 [31] which led to the introduction of the Information Society Programme in the 1990s. However, given the recent bruising from software opponents it is unlikely that the Commission will act further. This does not mean that software will not be protected throughout Europe, simply that in order to be protected the applications must be reconstituted in the requisite ('device style') manner. Some software patent opponents have welcomed the result of the European Parliament's vote, but it does not appear to me to be a particularly effective victory because clearly a flawed system with weak examination continues: the software patenting which existed before the Parliament's vote simply continues afterwards.

Some feel that other kinds of protection would be useful and more directed towards the special nature of software, for example, giving protection for a lesser period or providing protection more appropriate to SMEs. These frequently

[28] Beresford K., (2000) *Patenting Software under the European Patent Convention*, Sweet and Maxwell, London, p. 183. Emphasis added.
[29] Moor J.H., (1978) "Three Myths of Computer Science", *British Journal for the Philosophy of Science*, Vol. 29, No. 3, pp. 213-222.
[30] From an interview with Prof. Erich Hausser (then President of the German Patent Office). See Leith P., *Harmonisation of Intellectual Property in Europe: a case study in patent procedure*, Vol. 3, Perspectives on Intellectual Property, Sweet & Maxwell, London, 1998, p. 143.
[31] *Recommendations to the European Council: Europe and the global information society*, Brussels, 26 May 1994.

mirror the utility model and while, in some ways, they do make sense it is unlikely that they will encourage current patent applicants to give up seeking patent protection: more likely is that such utility models will be used to more effectively (and more cheaply) build a patent thicket around the more expensive patents.

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". [32] 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, there is a haphazard use of terminology in the field; there are also – to this author's eyes [33] – 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.

Board of appeal 3.5.1 is in the strongest position to improve what is – to most critics of the system – a confused and ill-constructed way to examine software given that despite the support of senior European judiciary for the European Patent Court, [34] the European Patent Litigation Agreement [35] is dead [36] and the Enlarged Board of Appeal is unlikely at this late date to tinker with notions of 'technical contribution'. [37]

---

[32] Garfinkel S.L., Stallman R.M., & Kapor M., (1991) "Why Patents are Bad for Software", *Issues in Science and Technology*, Fall, pp. 50-55.

[33] Leith P., (1990) *Formalism in AI and Computer Science*, Ellis Horwood/Simon and Schuster, London.

[34] www.eplaw.org/Downloads/Second%20Venice%20Resolution%20dated%204%20November%202006.pdf

[35] European Patent Organisation Working Party on Litigation (2005) *Draft Agreement on the establishment of a European patent litigation system* at http://www.european-patent-office.org/epo/epla/pdf/ewl0510.pdf

[36] Buck T., (2006), "Hopes fade for EU patents reform initiative", *Financial Times*, December 6. "… 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."

[37] The Enlarged Board has consistently refused to see itself as a upper level of appeal. Leith P., (2001) "Judicial and Administrative Roles: the patent appellate system in a European Context", *Intellectual Property Quarterly*, Vol. 1, 50-99.

Conclusion

The research project which underpins my argument was, in the simplest terms, to browse 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 affirmed it. My thinking about software patents changed and the conclusions I drew became more positive, though with some reservations.

These reservations are primarily of a technical nature: how do we examine software in a manner which will give protection for valid inventions (that is, those we presume to have a sufficient inventive step) and how do we find methods to bar those of the type which have brought software protection into disrepute. It seems unlikely that we can do this unless we think more clearly about the object of protection itself.

This is not to say that my own view is that software patents are either essential or 'bad for software' – the evidence is too weak to support either contention – but we are in a situation where software protection exists, will not easily go away, and thus needs to be dealt with as a currently existing yet damaged system.