Journal of Information, Law and Technology

The Proposed Software Directive: A User's Comments

Simon Davies
Patent Attorney, D Young and Company

Please note that views expressed here are purely personal, and do those of my employer or any other organisation I may be associated with.

This is a **commentary** published on: 4 July 2003.

Citation: Davies, S, 'The Proposed Software Directive: A User's Comments', 2003 (1) *The Journal of Information, Law and Technology (JILT)*.
<http://elj.warwick.ac.uk/jilt/03-1/davies.html>

1. Introduction

I have been asked to comment from the perspective of a user (i.e. industry) on the papers by Christian Koboldt and Robert Gehring. However, before doing so, I would like spend a little time discussing the background of the proposed Software Directive. This is important because one of the problems surrounding the Directive is that its purpose and intentions have been frequently confused. This is perhaps not entirely surprising, in that patents are a highly specialised field, and software patents a specialisation within this specialisation. It is therefore important to level set, in order to ensure that everyone is assessing the situation on the basis of the same (correct) factual matrix.

The legal provisions governing the patentability of computer program inventions in Europe are set out in the 1973 European Patent Convention (EPC). In particular, statutory subject matter is determined by Article 52 EPC as follows:

> Article 52 EPC
> 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;
>
> 1. The provisions of paragraph 2 shall exclude the 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.

Article 52 EPC therefore provides some form of exclusion of patentability in relation to computer programs, but the qualification of Article 52(3) makes the scope of this exclusion difficult to determine. The practice of the European Patent Office (EPO) has developed in a way that is generally quite favourable to software inventions. The tone of this approach was originally set by the **Vicom** decision regarding a method of two-dimensional data convolution, which was initially rejected as being a mathematical method and a computer program. On appeal however, the Board of Appeal held that claims restricted to the convolution of digital images pertained to real world physical entities, and so the invention could no longer be regarded as purely an abstract mathematical method. In addition, it would not be appropriate to deny patentability merely because the processing method happened to be implemented on modern

equipment, namely a computer system.

The general practice following Vicom and other subsequent EPO decisions has broadly treated computer program inventions as patentable subject matter, provided that they have some technical character/contribution/effect. This meant that all inventions that might reasonably be considered as within the realm of computer science, for example procedures at the operating system level to improve machine operation, or generic algorithms, techniques and functionality at the application level, would normally be regarded as outside the exclusion of Article 52(2) EPC. Such inventions could then be patented, provided of course that they meet the normal conditions for novelty and for inventive step, and also provided that they are claimed as a system and/or a method. Indeed, many thousands of software inventions have already been patented in Europe.

Another important milestone in the treatment of software inventions was a pair of **IBM** cases. In these cases the underlying inventions were clearly patentable according to the above criteria established by the EPO. However, the claims here were specifically directed to a computer program, rather than to a system or to a method. The Board held that such computer program claims were indeed allowable.

Note that a computer program claim may be expressed simply as a computer program for implementing the method of a preceding claim. If such a method claim is already regarded as patentable, then it is hard to see that also allowing the corresponding computer program claim represents a substantive difference in the fundamental nature of the invention being patented. However, superficially at least, it may be surprising that although the EPC clearly includes some form of exclusion for computer programs, it is now possible to obtain a patent with a claim that begins ‘A computer program....’.

This is not meant to suggest that the IBM decisions are incorrect, or that the general treatment of computer program inventions by the EPO is wrong. Rather, the EPC itself is unclear, and alternative interpretations are plausible. The EPO have adopted one particular interpretation, which is certainly a very reasonable one, but it is perhaps now impossible to know whether or not this was the approach that was originally intended when the EPC was first created.

The Commission became involved in the treatment of software patents back in 1997, with the Green Paper on the Community Patent and the Patent System in Europe. This document was primarily focused on the stalled Community Patent, but also addressed a couple of other issues. One of these was the need for harmonisation (often a prerequisite for the Commission to act), and the other was the rapid development of the so-called Information Society, triggered by the explosive growth of the Internet. The Green Paper posed two main questions in relation to software patentability. Firstly, it asked whether there was a need for harmonisation, in other words, was there any evidence for a lack of harmonisation. The second question was whether the exceptions to patentability in Article 52(2) EPC should be deleted.

In the subsequent Follow-Up to the Green Paper, the Commission appeared to have reached some firm conclusions. They noted the success of the US software industry, in

conjunction with the strong patent protection that had become available for software inventions in the US. The Follow-Up document stated that: 'the European Parliament supported the patentability of computer programs ... the Commission shares this analysis'.

The Commission also recognised that there was a problem due to the lack of transparency of the EPC. Thus as previously indicated, the treatment of software inventions by the EPO is not uniquely or unambiguously derivable from the EPC itself, but rather only one of several possible interpretations. They pointed out that this could cause problems if national Courts adopted a very different interpretation, since this would lead to a lack of harmonisation. In addition, the situation was clearly confusing to SMEs, who might naively interpret the EPC as simply prohibiting the patentability of all software inventions.

The Follow-Up document therefore proposed two 'urgent' actions. The first of these was for the Commission to prepare a Directive to ensure the uniform treatment of computer program inventions throughout the Community. The second proposal was that 'the Contracting States to the Munich Convention will … abolish computer programs from the list of non-patentable inventions'.

A diplomatic conference to revise the EPC was duly scheduled for November 2000. The Basic Proposal for revision going into the conference was to drop the computer program exclusion from Article 52(2) EPC. However, this turned out to be one of the few significant provisions of the Basic Proposal that was not incorporated into the final Revision Act adopted by the conference. It was indicated that this should not be regarded as a criticism of current EPO practice in relation to software inventions. Indeed, a decision not to amend a law is generally regarded as a confirmation of existing practice under the law.

There are probably two main factors behind why the Basic Proposal was not carried at the Diplomatic Conference. The first was lobbying by the Open Source movement, which is strongly opposed to software patents. Unfortunately, this lobbying is perhaps based on the mistaken premise that deleting the computer program exclusion would open the flood-gates to software patents (rather than simply represent a clarification of existing practice).

The second important factor was the significant change in practice in the United States regarding business method inventions. Thus in 1998, the Court of Appeal for the Federal Circuit (CAFC) in the **State Street Bank** case comprehensively rejected the notion that there was any prohibition on the patenting of business method inventions in the United States. This triggered a rapid expansion of activity in this field in the USA, and naturally led to discussion as to whether Europe should follow suit. Since many business method inventions are implemented on a computer, unfortunately this debate rapidly became mixed up with that on software patents.

The above factors led the Commission and the UK Patent Office to launch consultation exercises in late 2000 to try to determine what should be the policy towards both software inventions and business method inventions. It will be noted that the timing of these consultations overlapped the Diplomatic Conference to revise the EPC. This is probably

the primary reason why the amendment to Article 52(2) EPC was dropped from the Basic Proposal, since it was felt that it was premature to act pending the results of the consultations.

In fact it is debatable whether the consultation process threw up anything that was not readily predictable. There was broad support for software patents from industry, particularly from larger companies. The position of smaller firms was less clear, but this can easily be attributed to the confusion surrounding the debate. Typically such firms are uneasy at the (supposed) prospect of introducing software patents, but quite happy to continue with existing practice. In contrast to industry, software patents were strongly opposed by the Open Source movement. Perhaps the most useful conclusion to come out of the consultation was that no one in Europe was particularly supportive of the patentability of (non-technical) business methods. In other words, there was little or no pressure to move Europe in the same direction that the State Street Bank decision had taken the United States.

In the same time frame an important case went through the EPO, namely **Pension Benefits**, concerning an invention related to running a pension scheme using a computer. Although the method claims were rejected by the Board of Appeal as falling within the exclusion of Article 52(2)(c) EPC, the analogous apparatus claims for a system for implementing the business method were regarded as acceptable under Article 52(2) EPC, as being directed to a physical machine (i.e. a computer). However the apparatus claims were then rejected on the basis that the only advance was in an excluded area, in other words in a non-technical area, and so (apparently) could not contribute to inventive step. Accordingly, these claims were then rejected under Article 56 EPC as lacking an inventive step.

In February of this year, and after much internal wrangling (so it is believed), the Commission finally put forward the draft Directive. The Directive itself is rather short but is accompanied by a substantial amount of explanatory material. There were no great surprises in the proposals, which generally conformed to current EPO practice in terms of the treatment of software inventions. This included the adoption of the rationale of the Pensions Benefits decision as the basis for the treatment of (non-technical) business method inventions - these were to stay unpatentable (in line with the results of the consultation process). Perhaps the most worrying thing in the draft Directive from a user's perspective was the potential prohibition of computer program claims. Such claims provide a sensible and straightforward way to enforce patents for computer-implemented inventions, whereas their absence detracts from legal certainty. The adoption in the draft Directive of the rationale of the Pension Benefits case to reject system claims for business method inventions is also controversial, but beyond the scope of this paper.

2. Comments on 'Much Pain for Little Gain? A Critical View of Software Patents' by Christian Koboldt.

This paper attempts an economic analysis of software patentability. Although the general approach seems sensible and reasonable, I think that there is confusion at both the beginning and the end of the paper, which rather undermines its conclusions.

Thus the author initially complains that the Commissions proposals are 'a whimper rather than a bang', despite the Commission 'having identified a considerable need for action'. However, there is no indication of what form of 'bang' the author was expecting. Certainly, the Commission has never suggested that the planned Directive would take drastic action, and the proposals to confirm existing EPO practice were widely predictable, in view of the background discussed above. Indeed, the Commission generally had little choice in this regard, given the lack of compelling evidence to depart from existing practice, and to adopt a more liberal or more restrictive approach to software patentability.

Mr Koboldt then provides a general discussion of Intellectual Property Rights (IPRs) from an economic perspective. It has long been understood that these cut both ways. Thus if we go back to the inception of the patent system in Britain, namely the Statute of Monopolies of 1623, it will be appreciated that the general purpose of this Statute was to prohibit monopolies, since these were seen as detrimental to economic well-being. Nevertheless, a special exception was made in respect of granting a limited monopoly (i.e. a patent) to an inventor, on the basis that the disadvantage of such a restriction on third parties was outweighed by the benefit to society of encouraging innovation.

Mr Koboldt raises an interesting point regarding the application of existing IPRs, such as the patent system, to new technologies, for example, computer software. At a broad level we can identify two possible approaches to new technology:

(a) create an original, sui generis, right particularly directed to the new form of technology; or
(b) accommodate the new technology within existing IPRs.

I am tempted to say that the former approach is beloved of certain academics, who try to formulate IP schemes almost as an intellectual exercise. However, in practice there is no doubt that industry prefers the latter option. Thus it is generally far simpler to have a common approach across different technologies, since by utilising existing, well-understood IPR systems, a much high degree of legal certainty can be obtained (which is of paramount concern for industry). Naturally, this can sometimes lead to tensions within the patent system, as different technologies pull in different directions, but there is no sign yet of any major rifts appearing. The alternative, of creating a plethora of different IP rights for different technologies, is deeply unattractive. In the present instance, Mr Koboldt talks about trying to shoehorn computer software into the patent system. However, having worked with both computer hardware and computer software technology, my practical experience is very strongly that the patent system is just as appropriate and useful to the latter as to the former.

Mr Koboldt then goes on to discuss network effects in the software industry. It is important here to properly differentiate between the following two situations:

(i) the innovator obtains marketplace dominance simply by being the first entrant into the marketplace;
(ii) the innovator is unable to resist competition from an established company that is able

to exploit existing marketplace dominance.

It is a frequent contention that Microsoft Corporation, because of its dominance of the client platform, is now able to stifle innovation in this market sector.However, the patent system may in fact provide the ideal mechanism to encourage and reward the innovator in this situation, by providing them with a legal defence for their invention. Thus a patent can then be utilised to resist the marketplace control of a dominant player, and so ensure a fair commercial return for the innovator.

Under the heading, 'Is there a case for software patents ?', Mr Koboldt then postulates two main justifications for such patents:

(a) they provide a more effective form of protection than other IPRs (at least in some circumstances), and/or
(b) patent protection encourages more disclosure of technology than trade secret protection, and this in turn stimulates further innovation.

I think that most people would accept that these are indeed two of the main benefits of the patent system. However, as Mr Koboldt recognises, factor (b) by itself is insufficient, since unless it is attractive for a developer to patent his or her advance (i.e. unless factor (a) applies), such information will never be published to encourage innovation by others. In other words, the patent system is only useful where factor (a) holds.

Unfortunately, it is in the assessment of factor (a) that I feel Mr Koboldt goes somewhat astray. He acknowledges that 'seeking patent protection would appear to be attractive for those ideas where disclosure does not ultimately matter because they are obvious for all to see'. However, obvious in this context (i.e. not hidden within the product) does not equate to obviousness in the patent sense (i.e. whether or not a new idea is inventive).

Another area of confusion concerns interfaces. Mr Koboldt writes that 'ideas, principles and algorithms that are not obvious might still be better protected as trade secrets, in particular, if an effective regime for the compulsory licensing of interface information for patented innovations were in place'. I'm not aware that any such 'regime' is being considered, so the relevance of this observation is rather unclear.

The final conclusion of Mr Koboldt seems to be that patents are not an attractive option to developers, yet at the same time there could be a lot of them about, which would then act as a hindrance to further innovation ('much pain for little gain'). I'm not sure that this is a logically consistent position (why would they proliferate if they were not attractive?).

Crucially however, Mr Koboldt does not acknowledge that there are already thousands of granted software patents. Thus the simple and unavoidable truth is that for many developers, patent protection is indeed more attractive than a combination of copyright and trade secrets (in other words factor (a) applies). Moreover, these patents are not limited just to user interface inventions, or other areas where trade secret protection is not viable (although this clearly is one consideration in opting for patent protection). Rather, such patents embrace the whole range of computer technology.

There are a variety of reasons behind the attraction of patent protection. For example, it is unwise to place too much reliance on trade secret protection in an industry notorious for high staff turnover. In addition, technology licensing is very common in the computer industry, and patents provide a very efficient vehicle for enabling such licensing. (The presence of a patent reassures the licensor that its technology will not be stolen, and the licensee that it is obtaining something of genuine value).

In summary therefore, software patents are already woven into the fabric of the computer industry, and the proposed Directive does little more than confirm existing practice. This must surely be the sensible course, absent significant and compelling evidence to the contrary, and certainly there is nothing in Mr Koboldt's paper that could lead to a different conclusion.

3. Comments on 'Software Development, Intellectual Property, and IT Security', by Robert Gehring.

In this paper, Mr Gehring argues that security and (to a lesser extent) reliability weaknesses in existing programs can only be successfully countered via an Open Source approach. The patenting of software inventions then enters from two directions. It is argued firstly that the availability of software patents contributes directly to the initial existence of security problems. Secondly, such patents are a barrier to Open Source, and hence may prevent a solution of these security problems.

Mr Gehring starts by presenting a rather alarming portrait of the software industry - bugs and security holes everywhere. My initial rather cynical reaction to this is that if programmers have made such a mess of their own business, the last thing we want to do is to let them tell us how to run the patent system.

More seriously, in focussing on the issue of security, Mr Gehring has certainly (and no doubt deliberately) selected an issue of very significant concern. However, having adopted this as a problem to be solved, my impression is that Mr Gehring proceeds to advocate what appears to be a preordained solution (Open Source). Unfortunately, I remain sceptical that the desired nexus really exists. In other words, is there really good evidence that Open Source code will provide a cure for current security problems?

There are many reasons for taking this position. Firstly, Mr Gehring is perhaps rather selective in his discussion of conventional software development technology, since security concerns do in fact already receive considerable attention. For example, a security model is integrated into the heart of the Java platform from Sun Microsystems, Inc., which represents probably the most important and pervasive advance in software technology in the last decade. In addition, program vendors frequently advertise the security features of their products, and moreover, some independent certification procedures exist. - Accordingly, users are already able to make an informed judgement regarding the security capabilities of various solutions available in the marketplace.

I am not aware of any strong empirical evidence that the Open Source development

process is able to consistently develop more secure software than conventional processes. If you scan the Web you will find some Open Source code of the highest professional quality, but equally there is code that was probably put together by students after a long night drinking beer. Ultimately, the quality of code depends on its authors, and this is true irrespective of the development process.

Mr Gehring's paper does not contain anything to convince me to the contrary. Rather, the paper adopts what is perhaps an idealistic argument that Open Source code will have improved security due to the review and feedback of a multitude of users. However, the vast majority of commercial users have more urgent things to do with their business and/or lives than pore over the inner details of third party code. Instead, most people simply want code that provides the desired functionality out-of-the box. Ideally, if something does break down, they want to know that the supplier has a contractual obligation to fix it (not that they have to hack into it themselves).

This brings us to the heart of the problem, in that the two most important factors that compromise security are:

(i) the laxity of users in applying security updates, even where freely available, to their current systems, this being compounded by the fact that a network is only as secure as its most vulnerable component, and
(ii) a widespread homogeneity of software platforms (e.g. the pervasiveness of Microsoft products on the desktop), so that a single virus or other form of attack can disable a very high proportion of systems.

It is immediately apparent that Open Source cannot help address the first factor. Regarding the second, unfortunately Open Source programs are subject to the same homogenising pressures as conventional programs. If you have an Open Source operating system on your desktop, it is almost certainly Linux, while if you run an Open Source Web server it is almost certainly Apache. Of course, in theory the Open Source licensing does permit greater diversity, with each user customising their own system. However, bearing in mind the evidence from the first factor, namely that most users are too busy/unconcerned to apply even simple, readily available security patches, the prospect of them modifying their own source code for this purpose is exceedingly slim.

I should emphasise at this point that I do accept that Open Source code has important advantages from a customer perspective. For example, it provides some very high quality code, available at low or zero cost. In addition, major industry players back it. At the same time the customer avoids the risk of becoming locked into a single supplier. Unfortunately however, I just don't see improved security as one of its major selling points.

If Open Source code cannot resolve current security problems, then Mr Gehring's arguments against patent rights, as listed above, become largely redundant. Nevertheless, I would still like to review these arguments, since they do bring out some useful issues.

Thus Mr Gehring identifies two main problems with patent rights from a security

perspective, in particular:

(a) secure technology itself may be patented, thereby limiting its use; and
(b) binary distribution will be preferred, preventing proper understanding of security issues.

It should be realised that the first of these is simply part of the time-honoured balance of the patent process. In other words, third party rights are restricted (for a limited period) in order to improve innovation and progress for the overall benefit of society. Unfortunately, Mr Gehring does not seem to acknowledge the existence of this balance. Moreover, absent evidence to the contrary (and Mr Gehring does not provide any), the only logical assumption is that this balance falls the same way for security invention as for the rest of technology, in other words in favour of the patent system.

As regards the second point, this is split up into two factors: (i) trade secret protection is sustained, and (ii) the possibility of infringing third party rights makes it better to distribute in binary form. I do not understand the first of these, because the patent system in fact operates in exactly the opposite way to that suggested by Mr Gehring. In other words, as recognised by Mr Koboldt, patents are generally regarded as an alternative to trade secrets, and indeed it is a prerequisite of obtaining patent protection that the invention is publicly disclosed and described. Certainly therefore the patent system does nothing to encourage the use of trade secret protection, but rather would tend to reduce reliance upon it. A very large amount of technical information is published through the patent system.

Regarding the second factor, I accept that this could, in principle, motivate in favour of binary code. However, its influence is surely very weak, since distribution of binary code has been established as standard industry practice for very many years, and certainly pre-dates the widespread availability of software patents.

Turning now to section 6.2 of the paper, Mr Gehring talks more generally here about ‘the patent threat’. I'm afraid that this section repeats many common misconceptions about the current patent environment, such as that there has been ‘uninhibited patenting of nearly every trivial idea’. Thus quite apart from the standard tests of novelty and inventive step, patents are expensive, and this financial perspective is a powerful deterrent against over-patenting.

If we take a more general viewpoint for a general software developer, there is typically a choice between:

(i) in-house code development, which requires time and investment, but retains full IP rights; and
(ii) use of existing Open Source code (possibly with some modification). Here the investment is greatly reduced, and probably also time to market, but no IP is available (sometimes not even for the modifications).

It is not generally possible to run a development business on the basis of option (ii) alone

(i.e. the Open Source process), since the lack of IP rights prevents the creation of a revenue stream. Nevertheless, this can still be attractive in some circumstances, for example where the Open Source code may assist the sale of another product (for example, it allows the other product to interoperate with a new platform).

The key point is that industry wants both options to be available. Thus it does not want to be forced to always go down the second (Open Source) option, by undermining IP rights for software. On the other hand, it does not want IP rights to stifle Open Source development and creativity. It is clear therefore that a balance is required - and it also seems clear that the current balance is about correct. Thus the rather alarmist predictions from the Open Source community about the dangers of software patent rights are becoming increasingly untenable. Rather, it will be apparent to the impartial observer that, in practice, software patents and Open Source code are actually co-existing rather well.

Considering now the Conclusions and Recommendations of Mr Gehring, I find myself surprisingly untroubled by these (despite having disputed much of the rest of his paper). Thus the creation of a source code privilege, dependent upon the code in question being (i) in source form, rather than directly executable, and (ii) available free, with the right to distribute further, may indeed be workable. Such a source code privilege might then provide reassurance to Open Source developers, without undermining the commercial utilisation and exploitation of conventional IP rights. In summary therefore, although I disagree with how Mr Gehring arrived at the concept of a source code privilege, nevertheless I think that the idea is potentially meritorious, and certainly merits further consideration.

Notes and References