

Comments on the study *The Economic Impact of Patentability of Computer Programs*

Ed Avis, 48 Trinity Rise, SW2 2QR, UK. epa98@doc.ic.ac.uk

November 23, 2000

Abstract

These are comments I would like to submit to the Commission about its study on software patentability. The order of the comments follows the order in which the study is presented, which means that comments on the study's conclusions come first.

At the end are some more general comments on whether it is useful to grant patents on computer programs.

1 Section I

1.1 Open source development

I feel that the authors of the report have misunderstood the position of open source developers summed up in the quotation from one open source developer. This developer points out that while the platform itself is open, individuals are free to create proprietary products based on that platform. This is usually true, and it is what copyright achieves. But I know that most open source developers would not support the idea that a particular use of their platform could be patented by one company which then prevents others from making their own implementation—in effect making the platform itself proprietary, since it no longer provides an open base for competing developers to build on.

The anonymous developer comments: ‘The goal of open source is to make sure that IP rights or other proprietary rights do not interfere with that platform’. This is the case for copyright on software, but software patents often do interfere with a previously open platform and limit choice. This is particularly true given the broadness and triviality of many software patents currently being granted.

Actually, I feel that the emphasis given to open source developers is misleading. Certainly the individual developer is vulnerable to software patents, but then so are all development groups not big enough to fight back. It just happens that unpaid open source developers are at one end of the scale. (They are also more vulnerable because they cannot keep their development secret, a point I'll mention again later.)

1.2 Software patents necessary for American SMEs

Commenting on the situation in the United States, the authors write: ‘On the one hand there is abounding evidence that the profitability and growth of independent and SME software developers in the States has often been to a significant extent dependent on possession of patent rights’.

It should be clear that this does not imply that the existence of software patents helps small and medium-sized developers to grow. It could just as well be true that the legal climate is such that a small company without sufficient patents to use defensively is vulnerable to attacks from large patent-holding corporations, and thus it is necessary for small companies to acquire patents if they are to attract investment and grow. In fact I believe that this is the case. The fact that, in the present US legal climate, it is necessary for small firms to hire patent lawyers and start acquiring patents does not mean that small firms in Europe would be helped by the introduction of a similar system. It might just be a burden they would rather do without.

Also remember that a fast-growing, highly profitable company which holds many patents does not necessarily indicate a healthy marketplace. It could be that this company is doing well through blocking competitors from entering the market and through demanding licence money from other firms. Such behaviour does not usually help consumers, who would prefer an open and competitive market. It might be better to have several companies, not quite as fast-growing or as profitable, who are competing freely and growing the total size of this market segment faster than a single monopolistic firm.

It is jumping to conclusions to say, as the report does, that patentability of computer programs has helped the growth of SMEs in the US. All one can conclude is that in a market where your competitors are filing for patents, you need to do the same yourself if you expect to grow.

1.3 Do software patents help small companies?

The authors claim that lack of software patents in the early days of the computer industry made it easier for large companies to copy the ideas of smaller developers. But is there any evidence that widespread software patentability has stopped this trend? A small developer holding one or two patents on a key idea will not stand a chance in a legal battle with a large developer holding thousands of patents, many of them vague or extremely broad but nonetheless granted. Software patents certainly do not level the playing field in favour of the small developer—except perhaps for the company which does no development work itself (thus not infringing on anyone else’s patents) but distorts the market by accumulating patents on whole classes of computer programs.

(There is also the implicit assumption that it’s a bad thing for other companies to market their own products based on a particular idea. But in order for competition in software to exist at all, this must be allowed. The advantage of being first to market is particularly great in the software industry, there’s no reason to create extra barriers to entry for those who are trying to bring competing products to market.)

1.4 Invalid patents

The report acknowledges that many software patents granted in the US are invalid, but doesn't consider this important: 'Their existence... is hardly a significant barrier to software developers.'. This is to seriously underestimate the problem.

Even an infringement suit for an invalid patent requires hiring a patent lawyer and a trip to court. Even if you win, the process can be cripplingly expensive for a small software developer, impossible for an individual or start-up. (This is true even under a loser-pays system, since it can never be absolutely certain who will win.) Multiply that by the sheer number of vague, possibly invalid patents held by many large companies, each one of them the subject of infringement proceedings, and you will see that invalid patents are just as much a problem as valid ones.

The overheads of software development are much lower than those in most industries where patents are used; so the costs of fighting patent lawsuits are proportionally much higher. Any patents, invalid or otherwise, represent a real risk for the small software developer. Large companies can usually countersue with their own collection of patents and reach a cross-licensing agreement, so invalid patents aren't so much a problem for them.

1.5 Current quality of software patents

The report concludes that we need to make sure European patent offices (national and EPO) do not fall into the same trap as the USA, to make sure they do not grant 'obvious' patents. (It's not clear whether this means obvious to a software developer, or obvious to a patent examiner, which in practice is rather different.) There's no evidence that this is happening. If patentability is to be extended, it would be prudent to clean up the patent examination procedure before such an extension, rather than afterwards.

1.6 Antitrust law

Anti-trust regimes are not an effective way to stop companies abusing their software patent portfolio. The speed at which the courts work, and the speed at which the software industry works, mean that any possible action would come far too late and take far too long. It is not good to introduce one layer of litigation through software patentability, and then try to fix it with another legal process. Better to allow companies to write software without the threat of legal action hanging over them, and let them respond quickly to market demand without waiting for courts to make decisions.

1.7 Options presented for changing the law

On aims II and III, the report presents three options. I feel that these options are misleading and do not show the whole story.

Option 1: The report tries to present the status quo as being that computer programs are patentable, and suggests that the ‘computer programs as such’ exception be removed from the EPC. But this is not maintaining the status quo at all. The EPO has very cleverly managed to reinterpret the law to make an arbitrary and unclear distinction between ‘computer programs as such’ and ‘computer programs not as such’. This happened very recently in the lifetime of the EPC; if it were really the intention of the EPC’s authors, surely it would not take several decades for this new, enlightened interpretation to appear?

Since the EPO’s decision was not based on any economic criteria or any research of its effect on the software market, there is no reason to suppose that it represents any improvement on the previous, clear position that computer programs were not patentable.

The criterion of ‘technical effect’, which is supposed to distinguish between software as such and software not as such, does not have any clear meaning. Patents have been granted for purely abstract, mathematical algorithms such as data compression on the basis that this has a technical effect—the effect being that you might use such an algorithm to encode data travelling across a network, for example. But the same could be said of any algorithm to translate data from one format into another, so at a stroke all network protocols and file formats become patentable. Patents have been granted on user interfaces, so we can assume that any software whose purpose is user interaction is not software as such. So already all means of input and output are considered as having a technical effect; and for any other software patent application it is not difficult for a computer programmer to think up a possible technical effect which gets round the ‘software as such’ exclusion. For example, the EPC rejected a patent application for a homophone checker on the grounds that it did not have a technical effect—but conceivably one could use such a program to get more efficient data compression, thus putting it into the ever-expanding category of software not as such.

I’m sure you have seen the software patent ‘horror gallery’ listing very broad patents granted by the EPO on setting prices in a Website (WO9615505), dynamically generating web pages from a database (EP0747840), printing ingredients required for a recipe (WO9529453) and many others. Apart from the serious implications for competition, these should be sufficient proof that the notion ‘technical effect’ is in practice just used as an excuse to grant patents on as wide a range of programs as possible.

It’s hard to imagine the EPC ever ruling that something previously considered a technical effect is no longer to be counted as one—the border between patentable software and excluded software would move in one direction only. For these reasons, there is no real difference between Option 1 and Option 2.

But there is an option left out:

Option 0: Uphold the status quo as clearly defined by the EPC, making it clear that computer programs are not patentable, just as methods for playing games and methods for doing business are not patentable. This would not stifle competition and innovation in the software market; it would not expose small developers to legal threats; it would mean that companies can get on with writing software and not fighting court battles.

I urge you to consider this option too, and to recognize that the EPO's fig-leaf of 'technical effect' has no real meaning when applied to software.

2 Section II

Since I am not a lawyer, I will not write much on Section II, but I would like to say a few things about lawmaking:

Firstly there is the obvious point that there is a fundamental difference between a physical invention or process, and something which is purely information such as a picture or computer program. Making a clear, legally enforceable distinction between the two has already been managed by copyright law, so there is no reason why patent law cannot do the same.

But more importantly, the patent system exists only to serve economic interests. Any decision on what is patentable should be made on economic grounds and not by trying to carry across decisions made for a different area.

So removing exemption of computer programs on the grounds of 'technical effect' alone is not sensible, because having a technical effect and being economically justified are two unrelated criteria. What is worthwhile for other industries may not be worthwhile for software, and vice versa.

Finally, on the question of whether software is 'technology', I would like to point out that while marketing departments often refer to software as 'technology', you will not usually hear a programmer use this term.

3 Section III

Section III has a mixture of pro- and anti-swpat viewpoints. But those which favour software patents, or at least suggest that the anticompetitive effects they cause would not be too great, tend to be founded on the assumption that 'the system works properly'. So large numbers of bad patents are not being issued, companies cross-license on reasonable terms, antitrust laws work swiftly and effectively, and so on. But this idealized software patent world is not real. The software patents being granted by the EPO are frequently of very poor quality, just as bad as those in the US if not worse. And it only takes a single firm to start aggressively enforcing its software patents to cause serious trouble to every other firm which is not big enough to have its own defensive portfolio.

The claim that 'the existence of a patent system... discourages business secrecy' may be true in other areas, but not for software. Many companies cannot disclose their source code, even if customers want it, for fear that this would allow others to trawl through it for possible infringements. On the other hand, the typical software patent will not disclose anything of value; either it is too broad and vague to give any details, or contains ideas which would occur to many programmers thinking about the same problem. (There are some software patents which contain descriptions of new algorithms, but in these cases the algorithm would have been disclosed anyway without a patent, if it is to

become any sort of useful standard.) So software patents actually encourage business secrecy and discourage disclosure.

4 General comments on software patentability

The hard work of software development is usually not in thinking up an idea but in implementing that idea. This is particularly true given the obviousness of many of the ideas that the USPTO and EPO consider patentable, in some cases simply because they involve a computer. It is not like (for example) the drug industry where many years and lots of money must be expended to generate new patents. Rather, thinking up new ways to solve a problem is what a computer programmer does every day, and the same idea will almost certainly have occurred to many programmers in the past. The expense comes in implementing and testing code based on that idea.

The work of implementation is already protected by copyright, which provides a good balance between incentives for development and a competitive marketplace. Software patents tip that balance too far towards monopoly, and in any case it's unlikely that the first company to file a patent actually 'invented' the idea.

But even if, in theory, it were possible for the patent office to examine all the millions of lines of program code currently in use and grant patents only on genuinely new algorithms, the price would still be too great. The software market is unlike most others in its strong requirement for compatibility; competing products need to read each others' file formats, for example. A patent on a file format, or on an algorithm used to encode that format, allows one company to block all competing products that might perform the same function as its own.

Developers are expected to check every line of their code against thousands of existing patents. Consider that the USPTO has allowed the same algorithm (LZW compression) to be patented twice, by Unisys and by IBM. If even the patent office cannot check an application against previous patents, what hope is there for the developer checking a 500,000 line program? Copyright, on the other hand, does not have this problem; you have legal certainty that if you have not copied anyone else's work, you are not infringing. Copyright works well for computer software; patents do not.

In short, the possibility of infringing on thousands of software patents is a serious burden for small companies, and patents on file formats (or on business methods, which could effectively be granted if the business method involves a computer) affect even large companies. The result is reduced choice in the market and less software available to the consumer. And patents on file formats mean total monopoly with no possibility of competing, compatible programs being written.

The fact that American firms are stuck in this mess is not a reason to inflict it on European companies. I urge you to recommend that the existing wording of the EPC be upheld, so that computer programs (along with methods for playing games, doing business, and so on) are made explicitly not patentable.

Also, any change to the EPC which allows the EPO to decide for itself on changes to the rules would be very unwise. The patent office is not equipped to make a proper economic analysis of changes to the system. Past experience has shown that the EPO would move the frontier of patentability in one direction only.