Jnl. Intellectual Property Law and Practice Advance Access published January 10, 2006

The software patent debate

Andrés Guadamuz González*

It was never the object of patent laws to grant a monopoly for every trifling device, every shadow of a shade of an idea, which would naturally and spontaneously occur to any skilled mechanic or operator in the ordinary progress of manufactures. Such an indiscriminate creation of exclusive privileges tends rather to obstruct than to stimulate invention. It creates a class of speculative schemers who make it their business to watch the advancing wave of improvement, and gather its foam in the form of patented monopolies, which enable them to lay a heavy tax on the industry of the country, without contributing anything to the real advancement of the arts.

US Supreme Court, *Atlantic Works v Brady*, 1882

Key issues

- The recent demise of the proposed Directive on Computer-Implemented Inventions has overshadowed its complex background and the interplay of conflicting interests that it brought into play.
- Copyright (which already protects all software) and patent law (which protects much software in the United States but relatively little elsewhere) both have their strengths and weaknesses as legal rights. Little evidence has, however, been adduced as to the incentive effect of either of these legal regimes.
- The decision to withdraw the proposed Directive does not mean that the issues addressed in it and the interests affected by it have been resolved. Real debate has merely been deferred and it is important to recognize them clearly before the debate is resumed.

1. Introduction

The approval procedure of the proposed Directive on the Patentability of Computer-implemented Inventions[1] (the CII Directive) has sparked a heated debate regarding the patentability of software[2] in Europe, producing one of the most contentious intellectual property law policy discussions of recent years.

While the CII Directive has been rejected by the European Parliament,[3] the road that led to the final vote was paved with arguments and counter-arguments about the role that patents should play in the protection of software. This debate, which had strong political implications, was tinted by emotional appeals, threats, inaccuracies, and downright fabrications from both camps from the initial barrage. This article examines the arguments that are raised and considers their validity.[4]

Before looking at these arguments I will explain, through reference to some recent cases, the state-of-the-art regarding the legal protection of software. While the literature in this area is extensive and goes back several years, the time has come to take a step back and look at the debate again from its roots.

2. Software copyright: copy wrong?

Since it became widespread and commercially valuable, it has been remarkably difficult to classify software within a specific category of intellectual property protection. This is because characteristics of software are unique among protected intellectual creations, presenting particular difficulties for those

* Lecturer in E-Commerce Law, University of Edinburgh and Co-director of the AHRC Research Centre for Studies in Intellectual Property and Technology Law.

1 Proposal for a Directive of the European Parliament and the Council on the Patentability of Computer-Implemented Inventions, COM[2002] 92, http://www.europa.eu.int/comm/internal_market/indprop/docs/comp/com02-92en.pdf.

2 There are various definitions of software. That preferred here is US Code Title 17, ch 1, s 101 which defines a computer program as 'a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result'.

3 FFII, *European Parliament says No to Software Patents* (2005), http://wiki.ffii.de/Ep050706En

4 The author acknowledges that it is possible that the paper will not be free from some of his biases with regard to the validity of software patents. The author's position is of strong scepticism for software patents.

drawing analogies with existing legal subjects. Commentators have sought to classify it under copyright,[5] patents,[6] both copyright and patents,[7] trade secrets,[8] or even as a *sui generis* software right.[9] It is indicative of the complexity of the debate and the problems in defining the protection of software that while this issue has been the subject of discussion for more than 20 years recent developments suggest that there is still no solution in sight.

But what is it about software that makes its unequivocal classification so difficult? The problem may lie in the fact that software is not a monolithic work: it possesses several elements that could fall within different categories of intellectual property protection.

Software is not a monolithic work: it possesses several elements that can fall within different categories of intellectual property protection

If we define software as a set of instructions to a computer that bring about a certain result,[10] the manner in which those instructions are expressed should inform us about the type of intellectual property protection that applies. These instructions are initially expressed as source code—lines of instructions in a computer language. Because the source code is expressed in the written form, software may logically be defined as being subject to copyright protection as a literary work. This was the initial approach towards software protection in most of the existing legislation.[11] However, software is not source code alone; to be able to operate in a computer, software has to be translated into object code[12] by a process of compilation. This translation has no bearing on the type of protection awarded to the software because the object code is a direct result of the source code and should arguably be linked to its fate.[13]

A problem arises with the strict categorization of software as a literary work because software has other elements that may not be subject to copyright protection. Software is not merely a literary expression: its lines of code have a function that is independent of the grammatical construction of the lines of code. The source code of a computer program, while completely different from that of another program, may yet have the same function and produce a similar set of instructions that achieve a similar result. This is the basis of the idea/expression dichotomy[14] that is so frequently debated.[15]

Although there is some case law regarding literal infringement,[16] the courts have struggled with the non-literal aspects of software infringement. Is copyright infringed where the functional aspects of a computer program are copied? The answer has been a very complex and lengthy 'yes'. This is evidenced by the initial application of the idea-expression dichotomy to software,[17] then by the inception and reliance on the rather clunky doctrine of the so-called Abstraction-Filtration-Comparison[18] in the United States, which has been both applied and criticized by UK courts.[19] Most recently, protection of the functional elements of computer software has been revisited in the United Kingdom in *Navitaire v easyJet*:[20] a software company specializing in online airline booking software sued easyJet and software

5 J Dunn, 'Defining the Scope of Copyright Protection for Computer Software' [1986] 38 Stanford Law Review, 497.

6 E Gratton, 'Should Patent Protection Be Considered for Computer Software-Related Innovations?' [2002] 7 Computer Law Review & Technology Journal 2, 223.

7 R Widdison, 'Software Patents Pending?' [2000] The Journal of Information, Law and Technology 3, http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2000_3/widdison/.

8 DW Carstens, 'Legal Protection of Computer Software: Patents, Copyrights, and Trade Secrets' [1994] 20 Journal of Contemporary Law 13.

9 JC Philips, 'Sui Generis Intellectual Property Protection for Computer Software' [1992] 60 George Washington Law Review 997.

10 Definition used in n 2 above.

11 Harmonized in Europe through the Council Directive 91/250/EEC on the Legal Protection of Computer Programs, OJL 122/42. Also see Computer, Designs and Patents Act 1988, s 3(1) b.

12 Object code is machine-readable instructions that can be directly executed by the computer.

13 This idea is not new: note, 'Copyright Protection of Computer Program Object Code' [1983] 96 Harvard Law Review 1723.

14 For more about the dichotomy, S Ang, 'The Idea-Expression Dichotomy and Merger Doctrine in the Copyright Laws of the US and the UK' [1994] 2 International Journal of Law & Information Technology 2, 111.

15 eg D Luettgen, 'Functional Usefulness vs. Communicative Usefulness: Thin Copyright Protection for the Nonliteral Elements of Computer Programs' [1996] 4 Texas Intellectual Property Law Journal 233; LL Weinreb, 'Copyright for Functional Expression' [1998] 111 Harvard Law Review 1149.

16 Most recently *Cantor Fitzgerald International v Tradition (UK) Ltd* [1999] Masons CLR 157.

17 *Whelan Associates Inc v Jaslow Dental Laboratory Inc* [1987] FSR 1.

18 *Computer Associates International, Inc v Altai, Inc* (2nd Cir 1992) 61 USLW 2434. In short, this test abstracts all the elements found in the computer program, filters out the unprotectable ones and then compares what is left to search for similarities.

19 *John Richardson Computers Ltd v Flanders and Chemtec Ltd* [1993] FSR 497; *Ibcos Computers Ltd v Barclays Mercantile Highland Finance* [1994] FSR 275.

20 *Navitaire Inc v easyJet Airline Co* [2004] EWHC 1725 (Ch).

developers BulletProof, alleging that they had copied substantial functional elements from their reservation software. There Pumfrey J significantly diminishes copyright protection of functionality:

> Copyright protection for computer software is a given, but I do not feel that the courts should be astute to extend that protection into a region where only the functional effects of a program are in issue. There is a respectable case for saying that copyright is not, in general, concerned with functional effects, and there is some advantage in a bright line rule protecting only the claimant's embodiment of the function in software and not some superset of that software.[21]

It is precisely the difficulty in protecting literal and non-literal elements of software that has created the perceived need for the patentability of software, since patents protect the functional aspects of works. There is no idea/expression dichotomy in patent law. If an idea fulfils the requirements for patentability—patentable subject matter, novelty, and inventive step—it will be awarded patent protection.[22] American courts had already opened the door to the patentability of computer programs by allowing a patent for a software that controlled manufacturing processes as early as 1981.[23] Subsequent cases have expanded patentability of software in the United States.[24] With the patent door open, and the seeming chaos in the copyright protection camp, the subsequent explosion in successful applications by software companies in the United States was no surprise.[25]

3. The European Perspective

The road to the present

While the United States has been allowing practically unlimited patentability of software in recent years, Europe is following a different path, for two reasons. First, there is a clear-cut bias towards copyright protection through the Directive on the Legal Protection of Computer Programs.[26] Second, Article 52 (2)(c) of the European Patent Convention (EPC) specifically states that computer programs 'as such' should not be regarded as patentable subject matter.

However, practice and case law have allowed the limited patentability of the so-called 'computer implemented inventions'[27] that involve a technical effect (or contribution, or process).[28] These cases recognize a limited patentability threshold where an invention that will be implemented through a computer fulfils the requirement of technicality.[29] While the source code, or the literary and textual element of software, cannot be patented, software that produces some sort of effect in the same way that any other invention does will receive protection. Precise definition of this technical effect or process has been hard to pinpoint for more than 20 years since it was first enunciated by the Technical Board of Appeal of the European Patent Office (EPO).[30]

Decisions regarding software patentability follow arguments that resonate with those regarding the literal and functional protection of software in copyright. The EPO Board of Appeal shows this in *VICOM*, in situations where a computer process has a merely abstract and mathematical (unpatentable) effect are distinguished from those in which a computer process has a technical (one could read 'functional') effect and should therefore be subject to patentability.[31] Further cases, toying with this distinction, have offered reasoning that is often muddled or contradictory.[32] Nevertheless, a few principles can be gleaned from the existing case law. First, software 'as such', meaning programs 'considered to be mere abstract creations', remains unpatentable.[33] Second, the technical effect subject to the application must make a considerable contribution to the prior art. For example, *Merrill Lynch* says that 'There must be some technical advance on the prior art in the form of a new result.'[34]

21 ibid, para 94.

22 WR Cornish and D Llewelyn, *Intellectual Property: Patents, Copyright, Trade Marks & Allied Rights* (5th edn, 2003) 173–207.

23 *Diamond v Diehr* 67 L Ed 2d 155 [1981].

24 Amongst others *Paine, Webber, Jackson & Curtis Inc v Merril Lynch, Pierce, Fenner Smith Inc*, 564 F Supp 1358 [1983]; *In re Alappat*, 33 F. 3d 1526 [1994].

25 eg software patent applications had increased by 16% per year from 1986 to 1997; J Bessen and RM Hunt, *An Empirical Look at Software Patents*, Research on Innovation Working Paper 03-17/R (2004) http://www.researchoninnovation.org/swpat.pdf.

26 n 11 above.

27 The author finds that this legal concept has been hijacked as a euphemism for the more charged term 'software patent'.

28 *VICOM* [1987] 2 EPOR 74; *Merrill Lynch's Application* [1989] RPC 561; *Gale* [1991] RPC 305; *Fujitsu* [1997] RPC 608.

29 Widdison (n 7 above).

30 In *VICOM*, T208/84.

31 ibid.

32 Contrast, eg *VICOM* with *Fujitsu*.

33 *IBM*, T0935/97, 5.2.

34 n 28 above 569.

Nevertheless, even though most of the existing rulings share common elements, the real life application of these principles has been uneven in Europe, as is often the case with vague and ill-defined legal concepts. This lack of clarity prompted the European Commission to propose the CII Directive,[35] which was meant to overhaul European patent practices by making the wording of 'technical effect' more precise.[36] The proposed Directive contained its own definition of what constitutes a technical contribution, similar to the requirements of prior art encountered in the case law, stating that it meant 'a contribution to the state-of-the-art in a technical field which is not obvious to a person skilled in the art'.[37] What made this Directive controversial—and eventually spelt its downfall—was its approach towards the patentability of software 'as such', in contrast to technical effects. Article 5 clearly states,

> Member States shall ensure that a computer-implemented invention may be claimed as a product, that is as a programmed computer, a programmed computer network or other programmed apparatus, or as a process carried out by such a computer, computer network or apparatus through the execution of software.

As mentioned, the practice in existing cases was not to patent computer programs 'as such', which seemed to exclude computer programs that provided a process in itself that was not technical. The problem with Article 5 is that it opened the door to the patentability of software 'as such', leading to comments that it opened the door to American-style unlimited patentability of software.[38]

This is not the place to describe the tortuous process that led to the eventual demise of the CII Directive. Article 5 was, however, at the centre of an almost unprecedented display of lobbying and activism, triggered by an apparently straightforward directive dealing with the application and harmonization of some obscure legal technicalities which most people in the mainstream had never heard of. The proposal was met with vicious opposition from open source and free software activists,[39] whose opposition was echoed by some Parliamentary groups. The European Parliament was instrumental in the voting down of the Directive by eventually rejecting the Commission's text by 648 to 14 votes on 6 July 2005.[40]

Technicalities in 'technical'

The demise of the Directive leaves the issue of the patentability of software in Europe in the same situation as it was before the proposal in 2002. Since existing practice and case law still applies, it is important to consider them closely and try to gain an insight as to what may happen in the future.

The official line that only computer programs that contain a technical contribution will be patentable has been followed as well by the European Patent Office in their Examination Guidelines:

> If a computer program is capable of bringing about, when running on a computer, a further technical effect going beyond these normal physical effects, it is not excluded from patentability, irrespective of whether it is claimed by itself or as a record on a carrier[41]

However, the same Guidelines apparently recognize that the concept of technical contribution is problematic, recommending that examiners should first determine the novelty and inventive step of the claim before testing for technical contribution.[42]

United Kingdom Patent Office (UKPO) practice followed similar lines to the rest of Europe. However, the UKPO recognized that applicants and practitioners were confused about the technical contribution requirements, evidenced in their early consultation regarding software patents.[43] This uncertainty prompted the UKPO to organize a series of workshops in early 2005 in pursuit of a workable definition of 'technical contribution'.[44] Those attending the workshops were presented with several different

35 n 1 above.

36 A Duffus, 'The Proposal for a Directive on the Patentability of Computer-implemented Inventions' [2002] 16 International Review of Law, Computers & Technology 3, 331.

37 Art 2(b).

38 eg criticisms FFII, *EU Software Patent Directive Core Amendments* [2003], http://swpat.ffii.org/papers/europarl0309/cons0401/tab/index.en.html.

39 Some reasons for this opposition can be found here: A Guadamuz, 'Legal Challenges to Open Source Licences' [2005] 2 *SCRIPT-ed* 2, 163.

40 n 3 above.

41 EPO, *Examination Guidelines* (June 2005), s C, Ch 4, 2.3.6; s C, Ch 4, 2.3.6.

42 ibid.

43 United Kingdom Patent Office, *Should Patents be Granted for Computer Software or Ways of Doing Business?* (March 2001), http://www.patent.gov.uk/about/consultations/conclusions.htm.

44 United Kingdom Patent Office, *The European Computer Implemented Inventions Directive—Report on the Technical Contribution Workshops* [2005], http://www.patent.gov.uk/about/ippd/issues/eurocomp/full_report.pdf.

Table 1. Definitions A and B

Definition A (CII Directive)	Definition B (FFII)
'Technical Contribution' means a contribution to the state-of-the-art in a field of technology which is new and not obvious to a person skilled in the art. The technical contribution shall be assessed by consideration of the difference between the state-of-the-art and the scope of the patent claim considered as a whole, which must comprise technical features, irrespective of whether or not these are accompanied by non-technical features.	'Technical Contribution' means a contribution made by a claimed invention, considered as a whole, to the state-of-the-art in a field of technology. 'Technical' means belonging to a field of technology. New teaching about the use of controllable forces of nature under the control of a computer program, beyond the implementation of the data processing procedure itself, is technical. The processing, handling, representation, and presentation of information by a computer program are not technical, even where technical devices are employed for such purposes.

definitions of 'technical contribution'[45] and given three different types of case studies, to try to determine which definition fitted best. The case studies consisted of applications that, according to the UKPO, were (i) not patentable under the present law, (ii) patentable, and (iii) borderline or doubtful according to the existing guidelines.

The results showed that Definition A would result in considerably more patents than Definition B.[46] Many of the other definitions fell along that same axis, with two exceptions (Table 1).

These two definitions were less controversial and, according to the study, were 'well liked'. Definition F was minimalist: its shortness and elegance may have played a part as to why it was chosen by the participants. Definition L was penned by the UKPO as its interpretation of what the current cases in the UK require (Table 2).

The workshop's methodology is open to criticism on many grounds, including the reasoning by which some definitions were omitted.[47] Also, the willingness to reach a middle ground assumed greater priority than the need to assess the underlying reasons for the study. The report repeatedly comments on how restrictive the definitions are, thus assigning positive or negative values to them. A strong opponent of software patentability might think that the most restrictive definition was preferable to the most permissive one, and vice versa. By favouring Definition L, the study apparently concludes that at least many workshop participants are content with the *status quo* and are happy to rely on it. Is this a good thing?

Two recent cases assist in delineating the status quo. The first, *Halliburton Energy Services, Inc. v Smith International*[48] involved two technologies: a cone drill to dig for gas and oil and a software simulation program for designing the drill bits. Halliburton sued Smith, claimed that it held patents in both the drill and design software[49] and that Smith was using similar software to produce comparable results. Smith questioned the patents' validity. The design software patent contained a long technical description of drills and drill bits, and a description of the algorithm[50] used to design the software. The instructions to the person skilled in the art were extremely detailed and could only apply to that particular desired result. This seems precisely to be the type of patent that has a technical effect, however one defines it. Pumfrey J agreed[51] that there was nothing wrong with the patent *per se* and that it fulfilled the requirements of

45 Of which two were used in all the workshops, the definition from the CII Directive and a definition provided by the Foundation for a Free Information Infrastructure (FFII), one of the main critics of software patents.

46 ibid.

47 In particular, some of the tabled definitions from the European Parliament when criticizing the CII Directive. See the Consolidated version of the amended directive 'on the patentability of computer-implemented inventions', Europarl 2003–09-24.

48 *Halliburton Energy Services, Inc v Smith International* [2005], EWHC 1623 (Pat).

49 Patent EP1117894 for the software and EP1112433 for the cone drill.

50 An algorithm is a set of instructions for accomplishing some task which form one initial state to an expected result.

51 *Halliburton v Smith* (n 48 above) paras 215–218.

Table 2. Definitions F and L

Definition F	Definition L (Current UK Law)
A technical contribution should make a substantial, non-obvious advance to the state of knowledge in a technical field, and should not be representable as pure logic.	'Technical Contribution' means a contribution to the state-of-the-art in a field of technology which is new and not obvious to a person skilled in the art. The technical contribution shall be assessed by consideration of the difference between the state-of-the-art and the scope of the patent claim considered as a whole. In the case of a computer implemented invention the 'Technical Contribution' must be realisable within the apparatus on which it is implemented. A 'technical contribution' cannot occur as a result of the coding of an algorithm or lie within the program code. A program may only make a 'technical contribution' when the implemented function is used to contribute to the difference between the scope of the claim and the prior art. A 'Technical Contribution' cannot occur when the contribution lies solely in the application of a 'business method, mental act, or mathematical method'.

technicality.[52] In this regard Halliburton represents a perfect example of a software patent that contains an unequivocal technical effect.

The second case is *CFPH LLC's Application*.[53] CFPH applied for a patent with one claim (later divided into two applications).[54] The patent claimed a networked system for placing wages for current events in real time, where each event for which a wage was possible had a minimum wage amount. The system then checked the user's available credit and displayed only those events where the user could place a bet. The UKPO rejected the claim because it did not produce a technical effect and because it described a business method, which is also not patentable subject-matter.

On appeal, Deputy Judge Peter Prescott QC admitted having problems with the concept of technical contribution existing in practice and case law. Following a detailed analysis of existing cases, he offered a possible test for technicality:

> A patentable invention is new and non-obvious information about a thing or process that can be made or used in industry. What is new and not obvious can be ascertained by comparing what the inventor claims his invention to be with what was part of the state of the existing art. So the first step in the exercise should be to identify what it is the advance in the art that is said to be new and non-obvious (and susceptible of industrial application). The second step is to determine whether it is both new and not obvious (and susceptible of industrial application) under the description 'an invention' (in the sense of Article 52). Of course if it is not new the application will fail and there is no need to decide whether it was obvious.[55]

This test seeks to apply the EPO guidelines as far as possible, but emphasizing the case law and EPO decisions on the strict requirement of an inventive step that has a technical application.[56] Unfortunately, this analysis did not advance our understanding of 'technical', other than restating that the invention should have industrial application. The Deputy Judge was, however, correct to affirm that software, by the mere act of being software, should not be excluded from patentability: if it involves an inventive step,

52 The patent was, however, invalid for inadequate disclosure: according to the court, the threshold of disclosure is much higher in highly technical areas such as this: para 133.

53 *CFPH LLC Application* [2005] EWHC 1589 (Pat).

54 Applications GB 02268843 and 04193173.

55 CFPH Application (n 53 above) at 95.

56 ibid, at 94.

it could be subject to patent protection. As he concluded:

> '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.[57]

In the author's opinion, this presents the best test for patentability of software yet devised: the 'little man' test. If the software is immaterial to the claim, and if the software fulfils other patentability requirements, then the patent should be awarded. Given such a useful definition, it is strange that the UKPO has missed some of the finer points of this ruling in its new recommendations regarding patentability.[58] The new examination recommendations come in the aftermath of *Halliburton* and *CFPH*. While dealing with other considerations, they also comment on 'technical contribution'. While acknowledging the new two-step test in *CFPH*, the guidelines state that 'the change in approach does not change the boundary of what is patentable'. I beg to differ strongly with this conclusion.

4. To patent or not?

It is clear that the existing procedure is well established in favour of some limited patentability of software, even after the defeat of the Directive. European Commissioner Benita Ferrero-Waldner has pointed out that, despite the vote, 'patents for computer-implemented inventions will continue to be issued by national patent offices and the European Patent Office under existing law'.[59] This is an accurate statement: existing practice has led to a large number of European patents protecting processes found in computer software.[60] While the figures are far lower than in the United States,[61] the number of approved software patents is higher than would be expected in a region where computer programs are supposed to be excluded as patentable inventions.

It would be disingenuous to believe that the matter of software patents will be debated less in the coming years. If the practice is in favour of patentability, but legal definitions remain unclear, the time is ripe to question whether the patentability of software is itself beneficial. This section analyses the validity of some of the arguments for and against the patentability of software.

Arguments for patentability

A strongly compelling argument for the patentability of elements found in a computer program is similar to those arguments supporting the patentability of any other invention. If a computer program contains elements that meet patentability requirements, it should be awarded software protection. Since software development is a technical field like any other its results should be patentable.[62] This argument should be examined and expanded in light of the traditional justifications for the existence of patents in general.

Patents are commonly justified as a contract between inventors and society, where the former are awarded a limited monopoly for a period of time while the latter obtains a description of how others can work the invention.[63] If this argument is valid, society can only benefit from the patentability of some software inventions because the technology to work those ideas will be disclosed in the application, something that would not happen if the software was protected as a trade secret or under copyright. Copyright owners do not need to publish the source code, which makes working on the software more difficult. Some even argue that the disclosure element of

57 ibid, at 104.

58 United Kingdom Patent Office, *Patents Act 1977: Examining for Patentability*, 29 July 2005, http://www.patent.gov.uk/patent/notices/practice/examforpat.htm.

59 European Parliament, *Debate: Patentability of Computer-Implemented Inventions*, 6 July 2005, http://tinyurl.com/8op7z.

60 It is difficult to obtain actual data. According to FFII the EPO has approved more than 20 000 software patents (http://swpat.ffii.de/patents/stats/index.en.html). According to the United Kingdom Patent Office, of about 30 000 applications received each year, 20% are related to software, see http://www.patent.gov.uk/about/ippd/issues/cii-ukposition.htm.

61 In 1980 there were about 1080 software patents issued, while in 2002 there were 24 891 patents issued. Bessen and Hunt (n 25 above), p 47.

62 An argument often made by patent expert Greg Aharonian in his mailing list, PATNEWS. For example, PATNEWS 20050217 contains a discussion of why software is just a technical abstraction, and therefore, subject to patent protection.

63 JP Kesan and M Banik, 'Patents as Incomplete Contracts: Aligning Incentives for R&D Investment with Incentives to Disclose Prior Art' [2000] 2 Journal of Urban and Contemporary Law 23.

the patent system allows for more openness in the software development market.[64] This argument is compelling when one sees that the software industry requires openness and interoperability of standards in order to let programs interact with one another. However, openness can be obtained within a copyright-only framework of protection through the use of alternative development models such as open source software, where the source code is made available to the public, ensuring openness and interoperability.[65] Openness can also be obtained by the proliferation of non-proprietary standards and standard-setting bodies[66] that establish a common framework for development, which can be achieved without patents.

Another traditional justification of the patent system is that it serves as a just reward for the effort that has gone into the making of the invention: software should be no different in this respect. The reward in the shape of a patent serves as an incentive to innovators, as it can be argued that developers need means to recuperate their investment. Says Gratton:

> Incentive is important for software developers—to reward those who invest their time and money in technological invention and innovation, and thus to encourage such investments, has been the classic function of patents. In other areas of innovation, patents have encouraged substantial investment in research and development and have generally promoted innovation. There is no reason why the position should be any different for software developers or businesses.[67]

This argument would be credible were it not for the fact that it has been established in the literature that patents work really well as an incentive for innovation in some areas of technological innovation, but not in others.[68] Software development is a vibrant area of innovation, despite the uncertain nature of its legal protection. The success of open source software also serves to diminish the claim, as there is a field of endeavour where thousands of developers innovate without the incentive of patent protection.[69] Moreover, there is little direct evidence that software patents generate an incentive for innovation. In a report to the European Parliament, Bakels and Hugenholtz point out that there is not enough empirical evidence to demonstrate a direct causal relationship between innovation in the software industry and patents.[70] Correlation does not mean causation.

Software development is a vibrant area of innovation, despite the uncertain nature of its legal protection

Besides the traditional justifications for patentability, the main other argument for software patents has been the economic case. The patenting of computer inventions benefits large firms because they have the resources to apply for patents.[71] However, most of the literature defending computer implemented inventions argues that software patents also benefit small and medium enterprises (SMEs), because small and medium developers need patent protection if they are to enhance their profitability. A group of SMEs supporting the CII Directive gives several reasons why software patents are advantageous to their interests, including the failure of copyright to protect functional elements in software, provision of an incentive to investors, and better channels for earning profits from licensing.[72]

Problems with this position have been pointed out even in pro-patentability papers and studies.[73] A significant objection is that SMEs are extremely

64 BL Smith and SO Mann, 'Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?' [2004] 71 University of Chicago Law Review 241.

65 M Valimaki, 'A Practical Approach to the Problem of Open Source and Software Patents' [2004] 26 EIPR 12, 523.

66 For more about standard-setting bodies and IP, see RP Feldman, ML Rees, B Townshend, 'The Effect of Industry Standard Setting on Patent Licensing and Enforcement' [2000] 38 Communications Magazine IEEE 7, 112; C Shapiro, 'Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard-Setting' in J Lerner, A Jaffe and S Stern (eds), *Innovation Policy and the Economy* (2001), p. 119–50.

67 Gratton (n 6 above) at 251.

68 It has been well established that pharmaceutical and chemical industries are more likely to have innovations due to patents, while other fields are not as affected. E Mansfield, 'Patents and Innovation: An Empirical Study' [1986] 32 Management Science 175; and A Silberston, *The Economic Importance of Patents* (1987).

69 For more details about this, see S Weber, *The Success of Open Source* (2004).

70 R Bakels and B Hugenholtz, *The Patentability of Computer Programs. Discussion of the European-Level Legislation in the Field of Patents for Software* (2002) JURI 107 EN, p 17.

71 Gratton (n 6 above) at 249.

72 Computing Technology Industry Association, *SME Manifesto on Patents for Computer-Implemented Inventions* (April 2005), http://www.softwarechoice.org/download_files/SME_manifesto_0105.pdf.

73 RJ Mann, 'Do Patents Facilitate Financing in the Software Industry?' [2005] 83 Texas Law Review 961, 1009.

unlikely to rely on patents for protection of their software because of expense, fearing to enter into patent disputes with wealthier firms.[74] Furthermore, there appears to be little agreement even among SMEs about whether software patents are needed. Several studies have found a sharp divide between independent developers and some smaller businesses already established in the industry—those in an already advantageous position seem to be in favour of patentability, while smaller independent firms are against.[75] Similarly, a study to the European Commission dealing with the patentability of software in Europe comments:

> There is considerable evidence of concern by European independent software developers about the potential effects of patents on the development of computer program related inventions.[76]

An area in which software patents evidently prove to be an advantage is investment. A well-established link exists in the United States between intellectual property assets and investment in a business, particularly from venture capitalists.[77] However, if software patents allow investment, this benefit could be counterbalanced by software patentability's potential disadvantages, which will be analysed next.

Arguments against patentability

Reading some of the many websites that oppose software patents, one obtains a different picture of the ongoing debate. Many of these sites have no cohesive and relevant criticisms of software patentability in the European context. These sites offer considerable equivocation, misunderstanding, exaggeration, and even conspiracy theory, which do not assist the debate. One such site claims that software patents are pushed by greedy patent lawyers whose goal is to destroy copyright protection of software because copyright is free.[78] But that is not to say that well-stated and valid arguments against patentability do not exist.

One argument that carries more weight in the literature has been that software patents encourage the creation of the so-called 'patent thickets': a dense undergrowth of interrelated patents that researchers have to navigate in order to develop new technologies. There are two different types of thickets. The first one is a single technological innovation that may be protected by several patent holders. This situation would require anyone interested in developing software in that area to obtain separate licences from numerous owners.[79] The second type of thicket occurs when a product is covered by a large number of patents, not just one.[80] Patent thickets increase the cost of innovation, they encourage inefficiency through the creation of complex cross-licensing relations between companies, and they may even stop newcomers entering the market if they fail to penetrate the thicket. However, at least one commentator takes issue with critics of patent thickets: even where thickets exist, they have no effect on innovation through research and development spending.[81]

Another argument addresses the nature of software. If software has both functional and literary elements, the prominence of one of those elements as the software's defining characteristic should give us a better idea of how to protect it. As Eischen eloquently explains:

> Is software an act of engineering or communication? If software is a rational endeavor, improving quality involves better and more resources: better management, better tools, more disciplined production, and more programmers. If software is a craft, improving quality involves the exact opposite: focusing on less hierarchy, better knowledge, more-skilled programmers, and greater development flexibility.[82]

This argument is a key reason why discussion about the nature of software protection persists after

74 For more on this, see Bakels and Hugenholtz (n 72 above) 25; P Tang, J Adams and D Paré, *Patent Protection of Computer Programmes*, European Commission Report (2001).

75 PbT Consultants, *The Results of the European Commission Consultation Exercise on the Patentability of Computer Implemented Inventions* (July 2001), http://europa.eu.int/comm/internal_market/en/indprop/comp/softanalyse.pdf.

76 R Hart, P Holmes and J Reid, *The Economic Impact of Patentability of Computer Programs*, European Commission, ETD/99/B5-3000/E/106 (2000), p 3.

77 For a thorough study into this, see Mann (n 75 above).

78 eg the arguments presented here (http://www.nosoftwarepatents.com/en/m/dangers/index.html).

79 Shapiro (n 67 above).

80 J Bessen, *Patent Thickets: Strategic Patenting of Complex Technologies* (2003) SSRN Working Papers, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=327760.

81 Mann (n 75 above), 999–1004.

82 K Eischen, 'Software Development: An Outsider's View' [2002] 35 Computer 536, 38.

all these years. The problem is that each camp holds an entrenched view of what software is.

Some of the most vocal and active criticisms of software patentability lie within the Free and Open Source (FOSS) movements. A major plank upon which they base their opposition is that most open source licences are copyright licences. For example, of the 58 licences certified by the Open Source Initiative (OSI)[83] as complying with the open source definition, only the Apache Software License and the Open Software License contain clauses assigning patents owned by the licensor. The most used FOSS licence—the GNU General Public Licence (GPL)—goes further than providing a mere assignment, as it states in its preamble part of the case against software patents from the perspective of FOSS ideals:

> any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Until recently the GPL was considered to provide adequate protection against software patents by ensuring that a software project could not be hijacked by a patent owner and a wilful infringer of the licence.[84] This changed in recent years with the increase in software patent applications and, more importantly, with what FOSS developers perceived as a decay in the quality of patents granted by the USPTO. This reasoning suggests that the lack of patent quality means that patents are increasingly granted for processes and ideas that are obvious, do not represent an obvious step, or do not have prior art. As Bruce Perens says, 'the vast majority of software patents, some say as high as 95% of them, are actually invalid due to the existence of prior art'.[85] The result is a software environment polluted by bad software patents that affect open source developers who do not have the resources to defend themselves against allegations of infringement and cannot attempt to declare the patents invalid.[86]

From a European perspective, the arguments of the FOSS community against software patents are often informed by American practices. Many argue that the problem in the United States is not specifically with software patents but with the entire American patent system, bad-quality software patents being just part of the general lack of quality of patents currently emanating from the USPTO.[87] In particular Jaffe and Lerner warn that 'The real enemy of open-source software—and software innovation more generally—is the abysmal implementation of software patents, not the concept'.[88]

Is the situation only desperate for American open source developers, or is the same problem experienced in Europe too. So far, the fears of open source developers should be unjustified because Europe has not granted so many software patents. Nevertheless, there are enough examples of European patents that protect elements found in software that are not innovative and which in many instances have considerable prior art against them.[89] Another worrying example is that of the European LIBDCA open source software project, part of the VideoLAN project that produces the open source media player called VLC.[90] LIBDCA is just one of the components of the media player used for decoding a proprietary media format called Digital Theater Systems (DTS).[91] This format is protected by patent EP0864146 in Europe and US Patent 5,956,674.[92] DTS Inc, the owners of the patents, have sent a cease-and-desist letter to the LIBDCA project alleging infringement of their patent.[93] As a result the source code for the encoder had to be removed.

Cases such as LIBDCA illustrate a major problem concerning software patents that open source

83 Listed here (http://www.opensource.org/licenses/index.php).

84 J Lerner and J Tirole, *The Scope of Open Source Licensing* (2003) IDEI Working Papers 219, pp 13–14.

85 B Perens, *Software Patents v Free Software*, http://perens.com/Articles/Patents.html.

86 Bakels and Hugenholtz (n 72 above) 26–27. Also R Stallman, *Software Patents: Obstacles to Software Development* (25 March 2002), http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html.

87 According to Greg Aharonian, by 2003 there were 200 000 software patents, of which 120 000 are invalid (http://wiki.ffii.org/Greg040706En).

88 A Jaffe and J Lerner, *Innovation and Its Discontents* (2004), p 202.

89 Examples abound: a patent for the MP3 format (EP0287578); the infamous Amazon One-click patent (EP0927945B1); one that covers fuzzy logic operations (EP0488694); and one for object code applications (EP0527213). For more patents, see http://swpat.ffii.org/patents/samples/index.en.html.

90 For more details about the project, see http://www.videolan.org.

91 DTS, a proprietary multi-channel encoder that delivers high quality, low latency and high bitrate DVD audio.

92 The patent claims to protect an audio encoding method, which pretty much takes audio streams coming from one format and converts it into another one.

93 The letter can be found here (http://www.ffii.org/%7Ezoobab/libdts/).

advocates predicted. A patent owner threatens a small open source project, to which there is no other recourse than to cease the development of the software because the project cannot oppose the patent

It is difficult to justify protection for a field of endeavour in which significant innovation comes from developers who have no interest in obtaining patents

even if it suspects it could be invalid. Patent litigation is expensive and a small open source project, or SMEs developing proprietary software, often cannot afford to oppose patent claims against them. The grant of software patents makes open source developers believe that Europe has embarked on a slippery slope that will lead eventually to American-style patentability.

Despite the examples cited, most fears and concerns of the open source community have not yet come to pass. FOSS's development has blossomed despite the chaotic state of patentability of software in the United States and there has been no patent infringement litigation against any open source project.[94] However, the fact that the war has failed to materialise does not mean that it is not coming. Open source advocates are justified in being wary about software patents, but such fears should be proportional to the actual threat.

5. Conclusion

The demise of the CII Directive has left Europe in turmoil because the debate has not been resolved, only postponed. The EPO and national patent offices continue to struggle with the nebulous legal concept known as 'technical contribution'. Meanwhile one hopes that harmonized and rational practice will soon arise. Cases such as *CFPH* are encouraging in this respect.

Nevertheless, the law cannot ignore the wider policy issues at stake. The software patentability issue is difficult to resolve, considering the astounding diversity of opinions about the mere nature of computer programs. This being so, it is difficult to attempt to research the software industry in Europe and provide a balanced and measured study of the facts. Such a study is not feasible: research in this area will always have to deal with preconceptions and biases. Despite this, the main arguments offered by those who support wide patentability appear to lack the strength and decisiveness to eliminate all criticisms made by those against them. The situation is not as bad as the most belligerent websites and weblogs suggest. It is, however, clear that there are indeed some problems with software patents, particularly as a threat to those who have chosen the non-proprietary development route and are releasing their programs as open source software.

The author believes that there is insufficient evidence that software patents result in increased innovation. On the contrary, fears about a system paralysed by the fear of infringement are more likely to hold sway with those familiar with software development. It is difficult to justify protection for a field of endeavour in which a significant amount of innovation comes from developers who have no interest in obtaining and seeking patents. One should be wary of those who argue that the industry will collapse without patents; just using any open source software product will prove them wrong.

If we are still discussing the legal nature of software, perhaps the most obvious way forward is to resurrect the argument for creating a new type of protection. A sui generis software right could be the only solution to marry the functional and literal elements present in computer programs. But this discussion is an entirely different subject.

doi:10.1093/jiplp/jpi046

94 H Meeker, 'Open Source: The Sky Is Not Falling', *Linux Insider* (4 July 2005), http://www.technewsworld.com/story/44367.html.