scripted

Volume 6, Issue 2, August 2009

The Software Patent Thicket: A Matter Of Disclosure

Rosa Maria Ballardini[*]

Abstract

The high complexity of software products, as well as the increased number of intellectual property rights in the field, has created a dense thicket of overlapping patent claims that companies must navigate in order to operate in the sector. The lack of relevant prior art and the abstract nature of the software patent claims are the major causes of overlapping patents in the field. However, efforts have thus far been concentrated merely in improving the prior art repositories. The abstract nature of the patent claims and the disclosure concerns deriving from that have, however, not yet received sufficient attention.

This article pursues this subject, first by investigating the reasons for, and consequences of, overlapping IP rights in software-related patents. This analysis suggests that overlapping problems and, thus, the software patent thicket, cannot be effectively reduced unless issues related to abstraction and disclosure are addressed. On the basis of this, a more detailed description of the programme – including flowcharts, pseudocodes and, when necessary, parts of the source code – might be an essential requirement to be added to the description of the invention in natural language.

DOI: 10.2966/scrip.060209.207

[*] Researcher in IP law at HANKEN School of Economics; member of the INNOCENT Graduate School in IP law, IPR University Centre, University of Helsinki; visiting scholar at UC Berkeley, Boalt Hall (2008/2009). The author thanks Professor Niklas Bruun and Assistant Professor Marcus Norrgård for their valuable comments. Sincere thanks also go to the telecommunications and computing team of the Finnish Patent Office, the participants of the DFG Graduate School n. 1148, Intellectual Property and the Public Domain Seminar at the University of Bayreuth, and the 8th Intellectual Property Scholars Conference at the Stanford Law School for commenting on earlier drafts of this paper. The usual disclaimer applies.

1. Software-Related Patents[1] and Patent Thickets

In incremental component industries like computer software – where new developments necessarily build upon existing technologies and innovation occurs through small improvements rather than real breakthroughs – inventions are usually highly complex and often protected by multiple intellectual property rights. Increasingly, products incorporate not just a single invention, but rather a combination of many different components, each of which may be the subject of one or more patents. Consequently, numerous licences might be required to produce even a single commercial product.[2] Furthermore, the vast proliferation of software-related patents in a relatively short period, together with the lack of relevant prior art in the field and the special nature of software as an abstract technology, has led to unoriginal, obvious, and vague patents being issued. This horizontal and dense overlapping of multiple patent claims is termed the "patent thicket."[3]

The term "patent thicket" originates from a litigation case in the 1970s regarding Xerox's dominance of a portion of the photocopier.[4] However, the economist Carl Shapiro has recently reintroduced it in the academic discourse, defining patent thickets as:

> *A dense web of overlapping intellectual property rights that a company must hack its way through in order to actually commercialize new technology.*[5]

Software patent thickets are mainly caused by two sets of problems: overlapping problems, related to the 'quality' of the patents, on the one hand; and problems related to the vast number of patents issued on the other. Both concerns might appear during the life-span of software-related patents, although at different stages. Specifically, the former set of problems represents the direct cause of the latter one.

This article pursues the subject, first by investigating the reasons for, and consequences of, the software patent thicket. Both the European and American perspectives are considered. In particular, the US's software patent landscape serves as a background for the discussion on the European situation.

The general concern revolves around thickets and software-related patents, and not around patent thickets within the software industry *per se*. It is important to keep this distinction in mind because the majority of software-related patents are obtained by

[1] In this paper the same meaning is given to both the term "computer programme" and "software" in the context of patent protection. Although the actual significance of "software" and "computer programme" differs, in fact, these terms can be used interchangeably under the assumption that the difference between the two is clear to all parties in the discussion.

[2] M Lemley and C Shapiro, "Patent Holdup and Royalty Stacking" (2007) 85 *Texas Law Review* 7, at 1991-2050.

[3] C Shapiro, "Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard–Setting" (2001) 1 *Innovation Policy and the Economy*, at 118-150.

[4] *SCM Corp. v. Xerox Corp*, 645 F. 2d 1195 (2d Cir. 1981) and *In re Xerox Corp*, 86 F.T.C. 364 (1975).

[5] See note 3.

firms operating outside the software industry.[6] As a consequence, the problem of software patent thickets, as it is here described, might not necessarily be reflected in the software industry. This does not mean, however, that software-related patents do not contribute to patent thickets in other sectors. Indeed, many of the industries that obtain most software-related patents – like semiconductor and computer industries – have been clearly identified by many researchers as having patent thickets.[7]

This article argues that the lack of relevant prior art and, in particular, the abstract nature of the software patent claims are the major causes of the overlapping problems in the field. However, although efforts have thus far been concentrated in improving the prior art repositories and various projects have been launched in order to reach this goal, the abstract nature of the software-related patent claims and the disclosure concerns deriving from that have not yet received sufficient attention. The thesis pursued here argues that overlapping problems, and, thus, the software patent thicket, cannot be effectively reduced unless issues related to abstraction and disclosure are addressed.

On the basis of this, a more detailed description of the programme – including flowcharts, pseudocodes and, when necessary, parts of the source code – might be an essential requirement to be added to the description of the invention in natural language. Enhancing disclosure would reduce the abstract nature of the software patents claims, consequently providing support both for better defining the boundaries between patentable inventions and prior art; and for narrowing the scope of the patents granted. Increased notice would reduce the number of the applications filed and thus also indirectly smooth the thicket.

Such a policy might be essential, not only for patent officers to better evaluating the applications, but also to support judges in dealing with questions of infringement. Competitors and new inventors – who necessarily need full access to the patented inventions in order to improve upon it – would also clearly benefit from such a practice.

2. Overlapping Problems In Software Patents

Overlapping problems represent a major cause of the software patent thicket. For the purposes of the present article, overlapping problems refer to patents that either should not have been issued, because the invention was not new or not unobvious; or patents that, even if patented on inventions that fulfilled the patentability requirements, should have been granted a much narrower scope of protection. As is evident, overlapping problems might arise either during the examination phase or when a patent is challenged in courts where interpretation of the patent's scope is needed.

Probably the most memorable example of an unoriginal software-patent is the Amazon.com patent, granted by the USPTO in 1999.[8] The patent, nowadays known as

---

[6] J Bessen and R Hunt, "An Empirical Look at Software Patents", 16 *Journal of Economics and Management Strategy* 1, at 157-189.

[7] See J Bessen and M Meurer, *Patent Failure* (Princeton University Press, 2008), at 187-214.

[8] US patent 5,960,411. See also *Amazon.com v Barnesandnoble.com* [C99-1695P], Seattle District Court (12/01/1999).

Amazon's "1-click" patent, covered an online system that allowed customers to enter their credit card numbers and address information just once so that, on follow up visits to the website, all it would have taken to make a purchase would have been a single mouse-click. Although it is still not clear whether Amazon was indeed the first to implement such a purchasing process, it is evident that the idea behind the invention was extremely simple and obvious to the eyes of a person skilled in the art. This lack of inventiveness was, for instance, the basis of the revocation of Amazon's "Gift Order Patent" application by the EPO Opposition Division.[9]

Another remarkable example of "trivial" software patent is the Wang's patent, obtained in the US in 1988.[10] The patent claimed, among other things, the general use of "frames" to display different types of information retrieved from servers. Clearly this was a very broad patent that claimed a vast range of technologies that might use a graphical user interface – far beyond Wang's actual product. Immediately the issued-patent raised concerns among Internet companies. It was only following a lawsuit by Wang against Netscape that a court decision narrowed the scope of the patent by interpreting the term "frame" restrictively.[11]

Overlapping software patents are not, as is often thought, merely an American concern, but rather a global issue. Bergstra and Klint,[12] for instance, eloquently highlight how in Europe overlapping problems are also strongly felt in the computer programming sector.[13]

Within the examples, they cited a patent "Apparatus for handling tag pointers," granted to IBM,[14] and describing the addition of a tag-bit to pointers in order to discriminate them from ordinary data, which represented a clearly old technique used in various systems long before the filing of that patent application. Examples of this include a patent granted to Sun[15] for a method and apparatus for simultaneously displaying graphics and video data on a computer display, which was a common technique that had been in use for many years before the patent filing; and the patent "Data pre-fetch for script-based multimedia systems,"[16] granted to Intel in 2000, aiming at speeding up the execution of multimedia scripts running in a limited memory client by pre-fetching data references that occur in the script, which was an obvious, non-inventive technique.

As these examples show, Bergstra and Klint's study considered primarily applications from large companies such as Microsoft, IBM and Sun. Allegedly, these companies have large patent practices and sufficient resources to determine whether an invention is actually inventive and to identify prior art before filing a patent application. Thus it

[9] EPO opposition hearing regarding "Amazon gift ordering patent" EP 0927945 B1 (07 Dec 2007).

[10] US Patent 4,751,669: "Videotex Frame Processing."

[11] *Wang Lab, Inc. v America Online, Inc*, 197 F.3d 1377 (1999).

[12] J Bergstra, and P Klint, "About 'Trivial' Software Patents: The IsNot Case" (2007) 64 *Science of Computer Programming* 3, at 264-285.

[13] *Ibid.*

[14] EP 10186.

[15] EP 752695.

[16] EP 767940.

can be argued that the examples analysed by Bergstra and Klint represent potentially typical, rather than unusual, applications.[17]

It is evident that overlapping software patents might lead to several negative consequences – the "dense web" of intellectual property rights being the most direct one. Low patentability standards, in fact, inevitably lead to a vast number of patents being granted. The radical increase in the number of patents issued in the software field is clearly evidenced by recent statistics both in the US and in Europe.[18]

Other reasons of concern include uncertainty, both on the value of the property right and on the validity of the scope of the right;[19] high transaction costs, due to complicated licensing negotiations, the risks of "royalty stacking", "hold ups,"[20] and "blocking" patents;[21] and risks of infringement, litigation, patent trolls and threats of injunctions.[22]

To address the issues mentioned above, the thesis argues that overlapping software patents find justification mainly in the lack of good prior art repositories and, in particular, in the intrinsic abstract nature of the software patent claims. Both perspectives need to be taken into consideration in order to solve the problem.

2.1 Reasons For Overlapping Software Patents

2.1.1 Lack Of Relevant Prior Art

One of the main reasons for overlapping software patents is the lack of relevant prior art in the field. Judging the novelty and originality of the inventions without an adequate prior art search is clearly a highly challenging task. Two different problems should be distinguished: lack of knowledge due to the difficulty of gathering prior art material on the one hand; and lack of knowledge due to prior art not being publicly available on the other.

The first aspect refers to cases where the prior art, even when in printed form, is impossible or very difficult to be found. First, much of the prior art in software technology lies outside the areas where patent examiners traditionally look – i.e. printed resources, such as domestic and foreign patents, and published literature.[23]

[17] See note 12.

[18] For Europe see EPO Annual Report 2007 - Statistics (while the total number of patents granted has decreased with respect to the previous years, the fields of electronics and computing have continued to grow). See also "World Patent Report. A Statistical Review", WIPO (Ed. 2008).

[19] M Lemley and C Shapiro, "Probabilistic Patents" (2005) 19 *Journal of Economics Perspectives* 2, at 75-98.

[20] N Thumm, "Blocking Patents and Their Effects on Scientific Research: Evidence from the Biotechnology Industry" (2005) *IPR Helpdesk Bulletin*, No 23.

[21] R Merges, "Intellectual Property Rights and Bargaining Breakdown: the Case of Blocking Patents" (1994) 62 *Tennessee Law Review* 1, at 75-106; M Lemley, "The Economics of Improvements in Intellectual Property Law" (1997) 75 *Texas Law Review* 5, at 989-1084.

[22] Recent empirical studies in the US clearly show that litigation risks are particularly high in the software (and business methods) field. See note 7.

[23] J Park, "Evolution of Industry Knowledge in the Public Domain: Prior Art Searching for Software Patents" (2005) 2 *SCRIPT-ed* 1, at 47-70.

Some inventions, in fact, solely appear in textbooks or user manuals that are usually not available to the patent examiners.[24] Additionally, because multiple representations of the same inventions are available in software technology, it is highly challenging to draw the boundary between allegedly different inventions, as discerning which specific piece of prior art is relevant for the claimed invention might be very difficult.[25] The drawing of such a boundary, however, is essential in order to assess the inventions in terms of prior art. Finally, the use by many patent attorneys of very broad terms so as to avoid a patentability objection in software patent applications, has rendered it hard to find the pertinent pieces of prior art.[26] The abstract nature of software patent claims clearly exacerbates these challenges.

The second aspect of the problem relates to the fact that many well-known software techniques have never been disclosed, in printed form or at all. Although they might be used in the source code of many software systems, they do not appear in any scientific publication.[27] Moreover, some software inventions are merely incorporated into products. For instance, methods for doing business (which are most of the time implemented through the use of a computer programme) are not available in journals, libraries or by searching databases, but rather in the web-material and the business plans of companies.[28]

Particularly relevant is also the fact that software was not considered patentable subject-matter up until the end of the 1990s and, as a result, many software-related inventions were (and still are) protected by trade secrecy, and hence remain undisclosed. It was only after the *Diamond v Diehr* decision[29] in the US and the Vicom decision[30] at the EPO that computer programmes started to be partially accepted for patentability purposes.[31]

Some restrictions still currently apply in Europe, with the European Patent Convention (EPC) excluding computer programmes "as such" from the patentability field,[32] and the European Patent Office (EPO) accepting only "technical" inventions for patent law purposes.[33] It is worth mentioning that limitations to software

[24] *Ibid.*

[25] See note 7.

[26] M Lemley, "Rational Ignorance at the Patent Office" (2000-2001) 95 *Northwestern University Law Review* 4, at 1495-1532.

[27] See note 23.

[28] J Cohen, "Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of 'Lock-Out' Technologies" (1994-1995) 68 *Southern California Law Review* 5, at 1091-1202.

[29] *Diamond v Diehr*, 450 U.S. 175 (1980).

[30] T0208/84 *Computer Related Invention/Vicom* [1987] OJEPO 14.

[31] J Newman, "The Patentability of Computer-related Inventions in Europe" (1997) 19 *European Intellectual Property Review* 12; C Laub, "Software Patenting: Legal Standards in Europe and the US in View of Strategic Limitations of the IP Systems" (2006) 9 *Journal of World Intellectual Property* 3, at 344-372.

[32] Convention on the Grant of European Patents of 05 October 1975 (European Patent Convention), Article 52, paragraphs (2) and (3).

[33] See EPO, Guidelines for Examination, Part C, Ch 4 (2007).

patentability might also be imposed in the US following a recent Court of Appeal, Federal Circuit's decision in the *In Re Bilski* case.[34] Even though the decision mainly addresses business methods patents, restrictions to software patents in general might ensue depending on the interpretation of the "machine or transformation" test formulated in *Bilski*. The Supreme Court has now granted Bilski a writ of certiorari, thus more clarity on the proper application of the test is expected later in 2009.

Altogether, the features mentioned above have rendered it difficult to keep most established publications up-to-date with many new developments in the computer programming arena. Accordingly, patent officers have found it challenging to keep non-patent databases updated[35] and to find relevant prior art references when searching. Patent officers' ignorance of relevant patent literature has lead to many patents being issued on subject-matters already in the public domain.

2.1.2 Abstraction

Patent law traditionally does not protect abstract ideas or principles, but rather practical and tangible implementations that might derive from such ideas or principles. Software patents applications, however, often include abstract claims. This causes at least three major problems. Firstly, abstraction leads to technologies being claimed for that are often not yet known to the inventor at the time of the application. As a consequence, patents might be issued very broadly and inventors be rewarded for things they did not invent. Secondly, abstraction makes it hard for the examiners to judge over the sufficiency of disclosure of the inventions and thus to decide whether enough information has been provided to properly support the applications. Finally, such a configuration makes it difficult for patent officers to draw the boundary between allegedly new and inventive inventions, and prior art. The abstract nature of software patent claims, in fact, leads to the issuing of patents that do not possess clear boundaries and, thus, gives rise to high risks of infringement and opportunistic litigation.[36]

It is worth noticing that infringement risks do not only affect software companies. On the contrary, every company that uses a website, accounting database and some in-house software specialist to design and customise the system, can find themselves sitting "on the wrong side" of a patent infringement suit.[37] A clear example of such a configuration is *the Global Patent Holdings LLC v Green Bay Packers* case. Global Patent Holdings had a patent on the use of certain images on a web site. Global Patent Holdings sued companies such as CDW Corp, Motorola, the Green Bay Packers, Caterpillar, etc., seeking settlements between 7 and 15 million USD. None of these was a software company.

---

[34] *In Re Bilski*, F.3d, 2008 WL 4757110, 88 U.S.P.Q.2d (BNA) 1385 (Fed. Cir. Oct. 30, 2008).

[35] A detailed list of existing non-patent databases is available at the EPO, USTPO and JPO, see Park at note 23 above.

[36] See note 7 above, ch 9-10.

[37] *Global Patent Holdings LLC v Green Bay Packers, No 00 C 4623, and Global Patent Holdings LLC v CDW Corp.*, No 07 C 4476.

Litigation risks are particularly high in the United States, where recent studies have shown that software patents are responsible for a major share of the patent lawsuits, playing a central role in the failure of the patent system as a whole.[38]

In Europe, instead, the litigation activity still remains relatively low.[39] This is particularly due to the fact that no business methods *per se* and fewer weak software-related patents are granted in Europe. However, patent thickets clearly exist also in Europe and opportunistic behaviour might arise because of the currently unregulated use of patents. In particular, the creation of a common European litigation system under the proposal of the European Patent Litigation Agreement (EPLA),[40] or the Community Patent (COMPAT),[41] as it stands, raises serious concerns on the exacerbation of patent thickets. This issue is explained in more details later.

2.1.2 Software-Related Inventions v "Traditional" Inventions

The challenges raised by abstraction when assessing patentability of software-related inventions appear quite evident when comparing "traditional inventions" (i.e. physical and/or tangible inventions, closer to those that the patent system was originally designed for) with software-related inventions.

As an example of "traditional invention" let us consider a European patent granted by the EPO for a machine for cutting and splitting wood.[42] The machine takes timber wood as input and produces wood cut in pieces as output. The summary of the invention in the application stated:

> *A chopping machine for cutting and splitting a timber, said chopping machine comprising a crosscutting device for cutting the timber, a feeder for feeding the timber in its longitudinal direction to the crosscutting device, a splitting apparatus operated by a splitting cylinder for splitting a cut block of timber, said feeder comprising two elongated supporting surfaces forming a substantially horizontal trough open in the upward direction, into which the timber to be treated can be placed…*

[38] See note 7, at 120-146.

[39] See EU Commission, Internal Market, Patent Litigation Insurances Studies, at: http://ec.europa.eu/internal_market/indprop/patent/index_en.htm (accessed 25 Feb 2009). The study shows that in 2006 only four EU countries had had more than 100 patent cases (including all fields of technologies) per year, fourteen countries had had less that ten, and about half of the EU's member states had not had any case for many years.

[40] "Draft Agreement on the Establishment of a European Patent Litigation System," at: http://documents.epo.org/projects/babylon/eponet.nsf/0/B3884BE403F0CD8FC125723D004ADD0A/$File/agreement_draft_en.pdf (accessed 25 Feb 2009) and "Draft Statute of the European Patent Court", at: http://documents.epo.org/projects/babylon/eponet.nsf/0/885CCB85F5CC33ABC125723D004B15F9/$File/statute_draft_en.pdf (accessed 25 Feb 2009).

[41] For more information on the Community Patent see: http://ec.europa.eu/internal_market/indprop/patent/index_en.htm (accessed 25 Feb 2009).

[42] EP 1118438.

Let us now consider the computer-implemented invention claimed in *Vicom*.[43] In *Vicom*, the claims were both for a method of digitally processing images and for an apparatus for carrying out the method. The image was inputted as pixels and the processed image outputted as pixels. In other words, it was a digital signal processing system for image processing. The description of the invention mainly lied in Claim 1 and 8 (after the amendments filed on appeal), which read as follows:

> *1. A method of digitally processing images in the form of a two-dimensional data array having elements arranged in rows and columns in which an operator matrix of a size substantially smaller than the size of the data array is convolved with the data array...*
>
> *8. Apparatus for carrying out the method in Claim 1 including data input means for receiving said data array, and said data array to generate an operator matrix for scanning said data array to generate the required convolution of the operator matrix and the data array, characterised in that there are provided feedback means for transferring the output of the mask means to the data input means, and control means for causing the scanning and transferring of the output of the mask means to the data input means to be repeated a predetermined number of times.*

In the first example the machine had clearly physical implementations: the components were concrete (e.g. crosscutting device, feeder, splitting apparatus, etc.); and the functionality was easily comprehensible (also to persons not necessarily skilled in the art). Conversely, in Vicom, not only the did the components not have any physical implementation, but also the final result was intangible. Additionally, the language used in the application is clearly much more obscure. Overall, this configuration suggests that assessing patentability of software-related inventions might generally be much more challenging than for "traditional inventions."

Plotkin[44] also provides clear evidence of how assessing patentability of software-related inventions implies more interpretation than in other fields of technology. Plotkin addresses the issue by comparing software-related inventions with other electromechanical inventions in an attempt to demonstrate the extent to which the software's special qualities should be reflected in intellectual property laws.

He focuses in particular on the fact that patent law traditionally requires electromechanical device's components "to be conceived of, described, and claimed in terms of their physical, not merely logical, structure." In other words, for electromechanical devices, as well as a functional design, the provision of a physical structural design is hard and essential for obtaining patent protection. Conversely, in software, the logical structures described by the source code represent the end point of the invention. The step to their physical realisation is, instead, fully automated.[45]

[43] EP79300903.

[44] R Plotkin, "Computer Programming and the Automation of Invention: A Case for Software Patent Reform" (2003) 7 *UCLA Journal of Law and Technology* 2. Available at SSRN: http://ssrn.com/abstract=503822 (accessed 25 Feb 2009).

[45] *Ibid.*

The situation thus far described is worsened by the fact that software patent applications generally contain a higher number of total and independent claims than other types of patents.[46] Software patent applicants, in fact, are more likely than others to claim inventions in duplicative ways within a given patent. Software-related patents, for instance, often include sets of claims that characterise the invention as a method (or process), a machine or apparatus (or device), and a system. This is mainly because software inventions can be conceptualised in many different ways. The Blackboard patent on an "Internet-based education supporting system and method" granted by the USPTO[47] and recently litigated in the United States,[48] for instance, represents an outstanding example of such a configuration.[49]

The Blackboard's patent protected an "Internet-based education supporting system and method." In other words, it covered the entire e-learning process, including methods, virtual learning environments (VLEs), discussion forums, delivering classes via webcast or podcast, and just about everything else.

Considering the extreme broadness of this patent, it is not difficult to imagine that Desire2Learn, a direct competitor as well operating in the manufacturing of educational software business, was accused of infringement soon after the patent was granted. Subsequently, a jury in Lufkin, Texas found for the plaintiff on issues of literal infringement, infringement by equivalents, induced infringement and contributory infringement of the patent, and awarded Blackboard a total amount of 3,265,633.00 USD. A permanent injunction against Desire2Learn was then issued.

2.1.3 Conclusions

The analysis above shows that it is generally much more difficult to evaluate the concrete applicability of inventions containing abstract claims. Thus, more interpretation is required in order to assess the question of patentability. An extensive disclosure, therefore, would seem to be necessary in software patent applications in order to ease the process.

Surprisingly, however, disclosure in software-related patents is usually quite poor and these patents generally reveal very little of the inventions they protect. As a result, all the mentioned problems have been particularly accentuated – patents have been given too broad a scope of protection and patentees have been rewarded for things they did not invent. This has, overall, contributed to the intensification of the problems of overlapping patents in the field.

These aspects play a particularly important role for the purposes of the present article. As will be explained in the following sections, in fact, there is reason to believe that abstraction and, to a certain extent, prior art concerns could be reduced by promoting increased disclosure in the applications.

[46] See note 7.

[47] US patent 6,988,138.

[48] *Blackboard Inc. v Desire2Learn Inc.* [9:2006cv00155], Texas Eastern District Court, Lufkin Office (26 July 2006).

[49] *Blackboard Inc. v Desire2learn Inc.* [9:06CV155], Texas Eastern District Court, Lufkin Office, "Judgment and Permanent Injunction" (11 March 2008).

2.2 Disclosure Of Information In Software-Related Patents

2.2.1 The Requirement of Disclosure

The term "patent" derives from the Latin word patere which means "open letter."[50] In other words, it means to make the invention available for patent inspection. The requirement of disclosure in patent law, therefore, constitutes the tool to ensure the "openness" of the inventions.

Disclosure is a clear trade-off under which the imposition of a monopoly to society is justified by the assumption that the benefit deriving from it (i.e. further technological progress and innovation) is greater than the negative effects of those temporary restrictions.[51] Therefore, a patent represents both an incentive and a reward that the society gives for the disclosure of new information that might ultimately lead to more technological wealth.

In today's society the disclosure requirement is not only seen as a means to enable the public to reproduce the invention, but also as information that can be used to both improve on a specific invention and permit the conducting of further research in other fields of technology.[52] Additionally, by spreading technical knowledge, disclosure contributes to a more efficient allocation of resources.[53] Finally, a proper disclosure should enable the examiner to delimit clear boundaries within novel and original inventions, and prior art, which, in this way, should prevent the overlapping of patent rights.

In Europe, Article 83 EPC states that "the European patent application shall disclose the invention in a manner sufficiently clear and complete for it to be carried out by a person skilled in the art". This means that an application must contain sufficient information so as to allow a skilled man, using his common general knowledge, "to perceive the technical teaching inherent in the claimed invention and to put it into effect accordingly."[54]

Written description, enablement and best mode are all fundamental pre-requisites for obtaining a patent under US patent law. Section 112 of the US Patent Act, in fact, requires patent applicants to describe the invention in a manner sufficient to enable one of ordinary skill in the art to make it and use it, as well as setting forth the "best mode" to implement the invention.[55] Hence, generally speaking, sufficient disclosure plays a fundamental role for the issuing of "valid" patents. Some conditions specific to the software field, however, impose restrictions on the disclosure of software-related patents.

---

[50] G Friedrichsen, R Burchfield, and C Onions, *The Oxford Dictionary of English Etymology* (Oxford: OUP 1996).

[51] D Chisum et al, *Principles of Patent Law. Cases and Materials* (New York NY: Foundation Press 2004), at 49-51.

[52] *Ibid.*

[53] W Landes, and R Posner, *The Economic Structure of Intellectual Property Law* (Harvard University Press 2003), at 328.

[54] EPO Enlarged Board of Appeal Decision G 2/95 (21 Dec 1994) and G 2/98 (21 May 2001). See also EPO Board of Appeal Decision T1191/04 (22 Nov 2007).

[55] 35 U.S.C. §112.

2.2.2 The Software Patents' Restrictions

The special nature of software technology, as well as court decisions and legal dispositions, altogether have contributed to drastically curtailing the disclosure requirements in software-related patents. Disclosure concerns are particularly felt in the United States, where would-be patentees of software inventions are generally not required to disclose much. Case studies[56] have shown how a series of Federal Court decisions have drastically curtailed the enablement and best mode requirements by holding that patentees do not need to disclose source or object code, flowcharts or detailed descriptions of the patented programme, while finding high-level functional descriptions sufficient to satisfy disclosure requirements.[57]

In Europe, the EPC and its related dispositions impose a somehow higher disclosure threshold. Generally speaking, for computer-implemented inventions (CII) (that is, inventions for which the implementation involves the use of a computer, computer network, or other programmable apparatus – the invention having one or more features that are realised wholly or partly by means of computer programmes),[58] applicants must provide a description in natural language, whereas disclosing the invention through programming languages is considered insufficient.[59] In addition, applicants might provide flow diagrams and other examples to enable better understanding of the invention.[60] In Europe, however, neither the disclosure of the designing documents nor the source code of the programme behind the patented CII is necessary to fulfil disclosure requirements.[61]

Both in the US and in Europe, therefore, any obligation for patentees to disclose information of the programme behind the invention has been merely nullified. In such an abstract technology as computer software, however, this obscure configuration might be especially detrimental for promoting further progress.

Notably, competitors and new inventors need access to the programme (lying behind the invention) in order to reproduce the patented invention, build upon it and thus create further developments. As computer programming is a highly technical and difficult art, in fact, pretending that one with ordinary skills would be able to reconstruct a software invention, giving no more than the function the programme is

[56] D Burk and M Lemley, "Designing Optimal Software Patents", in R Hahn (eds) *Intellectual Property Rights in Frontier Industries: Software and Biotechnology* (AEI Press 2005).

[57] See, for example, *Fonar Corp v General Electric Co*, 107 F.3d 1543, 1549 (Fed. Cir. 1997) and *Northern Telecom, Inc v Datapoint Corp*, 908 F2d 931 (Fed. Cir.), cert. denied, 111 S. Ct. 296 (1990).

[58] See "Patents for Software? European Law and Practice" (2008) European Patent Office. Available at: http://documents.epo.org/projects/babylon/eponet.nsf/0/a0be115260b5ff71c125746d004c51a5/$FILE/patents_for_software_en.pdf (accessed 25 Feb 2009).

[59] *Ibid.* See also EPO, Guidelines for Examination, Part C-II, 4.15 (2007).

[60] *Ibid.*

[61] For instance, the EPO has recently issued a new brochure on the patentability of CII explicitly specifying that the disclosure of the inventive concept "does not require disclosure of a source code." See European Patent Office, "Patents for Software? European Law and Practice" (2009). Available at http://documents.epo.org/projects/babylon/eponet.nsf/0/a0be115260b5ff71c125746d004c51a5/$FILE/patents_for_software_en.pdf (accessed 11 June 2009).

to perform, seems highly unrealistic. However, since the source code is usually kept secret and programmers disclose very little information of the programme behind the inventions in the patents, reverse engineering might be necessary to reveal what the patentee has invented.[62]

Although this configuration is obviously not a unique problem of software-related patents, in such a field the consequences might be particularly detrimental due to the possible problems associated with decompilation.

2.2.3 Trouble with Reverse Engineering Software

Reverse engineering is an important practice to access technical information in many fields of technology. Inventors often need to reverse-engineer patented products in order to study and understand the technology so as to improve upon it. In the software field, reverse engineering through "decompilation" involves working backwards from the binary object code to produce a simulacrum of the original source code.[63] Although, theoretically, reverse engineering might be a successful tool to reveal the knowledge hidden behind patented software-related inventions, a series of both legal and technical reasons might block such an accomplishment.

From a legal perspective, software decompilation might be forbidden under certain conditions. Lemley and Cohen[64] have argued that because decompilation involves the generation of a copy of the patented programme, it might fall within the broad category of conduct prohibited by patent law – that is "making, using, selling and importing" the product. Specifically, decompilation might constitute "using" and "making" the patented programme by generating a temporary yet functional copy of it in the RAM memory.[65] On this issue, however, the European and American traditions differ.

In most European jurisdictions reverse engineering software through decompilation falls under the research privilege as "experimenting on."[66] Under the American practice, however, where reverse engineering is accepted only for non-commercial purposes, such an exemption might not apply. Nor would the doctrine of exhaustion suffice in solving the problem. The doctrine of exhaustion makes a specific distinction between "using and reselling" a particular copy of a patented product (permissible) and "making" a new copy of a patented product (not permissible).[67] The fact that decompilation constitutes "making" the patented programme might be found to be infringing patent law. The problem specifically arises with respect to product patents,

[62] See note 56. See also P Samuelson et al, "A Manifesto Concerning the Legal Protection of Computer Programs" (1994) 94 *Columbia Law Review* 8, at 2308-2432.

[63] A Johnson-Laird, "Software Reverse Engineering in the Real World" (1994) 19 U. Dayton L. Rev. 3, at 843-902.

[64] See M Lemley, and J Cohen, "Patent Scope and Innovation in the Software Industry", 89 *California Law Review* 1. Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=282790 (accessed 25 Feb 2009).

[65] *Ibid.*

[66] For more information see "Research Use of Patented Knowledge: a Review", STI Working Paper 2006/2, OECD Directorate for Science, Technology and Industry (STI).

[67] See note 56.

while for process patents the ordinary "use" of the process is justified either under principles of exhaustion or implied licence.[68]

Even when reverse engineering is considered lawful, such a process might incur technical problems that might render it highly inefficient.

Software is usually distributed in object code form. Decompiling object code is not only highly expensive and time-consuming, but it is also a very difficult and extreme way of examining the structure of software applications.[69] This is particularly felt in modern software technology where specific tools are commonly used to make reverse engineering very challenging. Furthermore, the fact that compilation from source code to binary code always loses information and comments written by the programmers, often renders decompilation quite useless.

Overall, it can be concluded that reverse engineering should not be a necessary tool to replicate software-related inventions. Firstly, patent disclosure per se does not imply access to any artefact in order to reveal the secrets behind patented inventions. Secondly, reverse engineering is not a very useful tool in software: on the one hand it merely enables the understanding of the implementation, not the design, of the computer programme, which is better conveyed through higher-level means; on the other, the more developments rely on sophisticated tools to produce code automatically (different tools producing different codes for the same specification), the less useful reverse engineering becomes.[70]

2.2.4 Conclusions

The intangibility of software-related inventions, as well as the difficulty to discern information from those types of patents, call for a more extensive disclosure policy. On the one hand, enhanced disclosure is important in order to reduce one of the major causes of overlapping software patents, i.e. the abstractness of the claims; on the other, it might also be essential for software-related patents to meet patent law purposes. Patents are a quid pro quo under which the imposition of a monopoly is justified by the assumption that the society would be better off under this restriction due to the benefit deriving from the disclosure of new technological information. On these grounds, the failing to meet a sufficient 'disclosing threshold' might well be considered unlawful.

3. Trends In Legislation: An Overview

As technology becomes increasingly complex the patent thicket is accentuated. This trend is particularly reflected in highly technical fields, such as computer programmes.

---

[68] *Ibid.*

[69] See A Johnson-Laird, "Reverse Engineering of Software: Separating Legal Mythology from Actual Technology" (1992) 5 *Software Law Journal* 2, at 331-354. See also P Samuelson, and S Scotchmer, "The Law and Economics of Reverse Engineering" (2002) 111 *Yale Law Journal* 7, at 1575-1664.

[70] See M Campbell-Kelly, and P Valduriez, "A Technical Critique of Fifty Software Patents" (January 2005). Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=650921 (accessed 25 Feb 2009).

Recently, legislators on both sides of the Atlantic have started to realise the potentially devastating effects patent thickets can bring to innovation, and awareness has been raised at the patent offices and the national courts. This trend highlights that the software patent thicket is clearly a global concern and, as such, global solutions rather than national-based proposals should be promoted.

It should be stressed that, even though problems related to patent thickets in abstract technologies have now been openly recognised, attention has thus far been posed merely on issues of patentable subject-matters, novelty, non-obviousness and use of injunction. The disclosure requirement, instead, has not yet been properly considered as a remedy, neither by the courts nor by the legislators.

3.1 US Developments

In the United States, the problem of patent thickets has recently caught the attention of much of the scientific and engineering community in a number of technological arenas.[71] The 2003 Federal Trade Commission (FTC) report,[72] for example, emphasises how in certain industries the large number of patents issued makes it virtually impossible to search all the potentially relevant patents; review the claims contained in each of these patents; and evaluate the infringement risks or the need for licences. The Commission cites hold–up problems for the software sector, in particular, by pointing out that "the owner of any one of the multitude of patented technologies constituting a software program can hold up production of innovative new software."[73]

This wave of criticism led both the legislator and the courts to initiate different plans of action to tackle the problem. This is reflected, for instance, in the Strategic Plan[74] launched by the USPTO to improve the general patent landscape, and on the heated debates on issues of patent reform. Additionally, there has been a renewed interest by the courts in general, and by the Supreme Court in particular, towards patent cases.[75] The need to both curtail patentable subject-matter and curb patent rights emerges from some recent decisions in the software and business method patents sector.

eBay v MercExchange,[76] for instance, was a business method case which limited the use of injunctions in infringement cases and, thus, represented a clear effort of the Supreme Court to tackle one of the most direct consequences of patent thickets, i.e. patent trolls.[77] The *KSR Intern. v Teleflex* decision[78] is another clear attempt by the

[71] See G Clarkson, "Cyberinfrastructure and Patent Thickets: Challenges and Responses" (2007) 12 *First Monday* 6.

[72] See also Federal Trade Commission, "To Promote Innovation: the Proper Balance of Competition and Patent Law and Policy. A Report by the Federal Trade Commission," October 2003, Ch 3-V.

[73] *Ibid.*

[74] United States Patent and Trademark Office, 2007-2012 Strategic Plan. Available at: http://www.uspto.gov/web/offices/com/strat2007 (accessed 25 Feb 2009).

[75] See T Holbrook, "Return of the Supreme Court to Patents" (2007) 1 *Akron Intellectual Property Journal* 2.

[76] See *eBay v MercExchange* 126 S.Ct 1836 (2006).

[77] After the eBay decision, in fact, injunctions have been denied in cases where the patent holders did not manufacture or market the invention, or where the patent holder had already licenced the invention

Supreme Court to reduce patent thickets by affording more power to the examiners to reject applications on grounds of obviousness. *In re Comiskey*,[79] a decision on a software-related invention passed down by the Court of Appeal, Federal Circuit, highlighted and further discussed the *KSR*'s principles on the application of the non-obviousness requirement.

The discussion over patentable subject-matter has also been actively reintroduced in the discourse. Although the interpretation of the excluded categories from patent law has been the subject of debates for scholars, legislators and courts for a long time, this issue appeared to have been settled in 1998 with a Court of Appeal, Federal Circuit's landmark decision in the *State Street Bank* case.[80] By affirming that even though "mathematical algorithms are not patentable to the extent that they are merely abstract ideas," when such algorithms produce a "useful, tangible and concrete result" no reason should impede them to be considered patentable subject-matter. In fact, State Street officially opened the doors to both software and business methods patents in the US.

However, despite the fact that *State Street* has been a major cause of the increase in the patents filed and issued, as well as of the patent quality concerns in the fields of software and business methods, the Federal Circuit has now put forward a new line of interpretation, based on a more restrictive approach. The recent *In Re Bilski en banc* decision[81] and its "machine-or-transformation" test, represent a clear attempt to pose limits to the patentability of software and business methods inventions.

3.2 European Developments: What Europe Is Doing Wrong

While the American legal community has evidently started to realise the negative effects of software patent thickets and is embracing a more restrictive approach accordingly, the recent trends in European legislation as well as the EPO case law seem, instead, to be following a different track – with the EPO surprisingly lowering the bar for the issuing of software-related patents.

On the legislative side, harmonisation projects of patent laws – such as the COMPAT and the EPLA, as they currently stand – pose serious concerns towards the potential increase of patent thickets in Europe. On the one hand, the creation of a unitary litigation system is highly desirable in Europe. Under the existing rules of national enforcement, in fact, litigation costs are often very prohibitive. Additionally, this situation creates fragmentation, distortion of the internal market and uncertainty, making it very challenging for companies, users, courts and legislators to operate in the system.[82] On the other hand, the creation of a unique European patent court to deal

---

to others. See, for instance, *z4 Technologies, Inc. v Microscop Corp*, 434 F. Supp. 2d 437, 440 (E.D. Tex. 2006); *Paice LLC v Toyota Motor Corp*, 2006 WL 2385139, 5 (E.D. Tex. Aug. 16, 2006).

[78] *KSR Intern. Co v Teleflex Inc*, 127 S.Ct. 1727, 1739 ff. (US 2007).

[79] *In re Comiskey*, Fed. Cir. 2007-1286.

[80] *State Street Bank & Trust Co. v Signature Fin. Group Inc*., 149 F.3d 1368, 47 U.S.P.Q.2d 1596 (Fed. Cir. 1998). See also AT&*T Corp. v Excel Communications, Inc*., 172, F.3d 1352 (Fed. Cir. 1999).

[81] See note 34.

[82] For more information see R Ballardini, "Software Patents in Europe: the Technical Requirement Dilemma" (2008) 3 *Journal of Intellectual Property Law & Practice* 9, at 563-575.

with patent issues all across Europe raises patent "quality" concerns, as well as anxieties over the effectiveness of the future patent examination system. This last aspect, in particular, stresses the need to find a workable solution to patent thickets, in particular in the ICT field, where the problem reaches its peak. Neglecting this important step could lead to wrong policy decisions as well as inefficient solutions.

An example is the position recently supported by some of the major European ICT companies, such as Nokia, to oppose important harmonisation processes like the COMPACT and the EPLA, as they fear that the harmonisation on the litigation side would inevitably degenerate into a "charter for trolls."[83] Additionally, the unusual proposal put forward by IBM – to eliminate injunctions on patents that would be issued at European Community level on the framework of the Soft-IP project[84] – reflects this perspective.

However, it appears clear that the currently inefficient European litigation system would strongly benefit from harmonisation. It is also evident that harmonisation is not the cause of, nor would function as a direct aggravation to, the patent thickets or patent trolls in Europe. Instead, what this analysis most certainly highlights is that software patent thickets have their roots in very different reasons that thus need to be addressed and solved in the first place in order to create the proper basis for a harmonised European litigation system to function properly.

On the examination front, the most recent EPO case law clearly shows how the requirements needed to grant computer-implemented inventions patents are being increasingly relaxed.

It should be noted that the fact that computer programmes – "as such" – are excluded under Article 52(2)(3) EPC, does not mean that software-related patents are never granted in Europe. Specifically, the EPO grants patents on CII as far as they have technical character, are new and involve an inventive technical contribution to the prior art.[85]

The "any hardware" approach nowadays followed at the EPO,[86] however, radically limits the "as such" exclusion of computer programmes from European patent law. On the one hand, under the "any hardware" approach it is easy enough to satisfy the "invention = technical" test – it being sufficient to refer to some features of the hardware in the patent claims. The technical nature of the invention is, instead, assessed while examining the inventive step. In other words, under this practice, the non-patentable method (i.e. the computer programme) should be blocked at the inventive step examination, as no unobvious technical solution is provided to solve a technical problem. On the other hand, however, some recent case law[87] shows that

[83] See, for instance, E Hakoranta (Director, IPR Legal, Nokia Oyj), "Market for Patents. The Future of Fragmented IP & Boundaries of Patent System Explored" (2008), Current Trends in Patent Law Seminar, IPR University Center.

[84] J Sage, "Soft IP", Computer-Implemented Inventions: Where Do We Stand in the Debate on "Software Patents"?, Brussels 5

[85] See note 61.

[86] T 0931/195 *Controlling Pension Benefit System/PBS Partnership* [2001] OJEPO 441 and T 0258/03 *Auction Method / Hitachi* [2004] OJEPO 575.

[87] T0424/03 *Microsoft / Data transfer expanded clipboard formats* [2006].

this latter requirement might also be overlooked by the EPO. In this way, the seemingly absolutist "as such" exclusion appears to have lost any substantial meaning under current rules. The overall general increase in the number of software-related patents issued by the Office during the past ten years is the direct consequence of this trend.

Overall, this analysis suggests that, even though software patent thicket problems might still be relatively low in Europe if compared to the US, these concerns are likely to increase if immediate actions are not taken both on the legislative and jurisprudential side.

4. How To Improve The System?

From a theoretical point of view there are different solutions that might be adopted to clear the software patent thicket. Scholars, courts and legislators dealing with the issue have so far mostly concentrated on two of these types: eliminating patent protection for software,[88] on the one hand, and curtailing the patentable subject-matters[89] on the other.

Some have also focused on the proper application of the patentability criteria as a "passing filter" for valid patents. [90]Even in this latter case, however, attention has mostly been posed on the novelty and non-obviousness requirements. Disclosure matters, instead, have not yet been sufficiently considered. As a consequence, disclosure obligations remain remarkably low under current rules.

4.1 The Story Thus Far

As already mentioned, one of the proposals put forward in order to solve software patent thickets is the complete elimination of patent protection. In fact, ever since software has been considered patentable subject-matter, some supporters of the doctrine have been lobbying against the affording of a twenty year monopoly to such a dynamic, fast-changing field of technology. To give fuel to their arguments, scholars in various disciplines have also tried unsuccessfully to find a clear-cut answer to the question of whether patents promote or hinder innovation in the software field, in order to decide whether software should or should not be protected by patent law. Thus far, however, incentive theories in specific technological fields have been very difficult to support with empirical evidence. The incentive question, in fact, appears to be a matter of patent law in general, rather than of specific individual sectors of technology.

These questions still remain extremely challenging and finding an answer goes beyond the scope of this paper. Nevertheless, there seems to be strong evidence that banning patents for software would most probably be unrealistic: patent protection for computer programme applications exists and is clearly here to stay. Thus, it needs to be dealt with as an imperfect system that exists today.

---

[88] R Stallman, "The Danger of Software Patents" (2005). Available at: http://notabug.com/2002/rms-essays.pdf (accessed 25 Feb 2009).

[89] For example, *In Re Bilski,* note 34.

[90] See note 64.

Questions on software patentability under the "patentable subject-matter" doctrine have also been a matter of discussion among scholars, patent officers, legislators and judges for a long time. The impossibility of limiting the type of admitted claims in software-related inventions, however, is eloquently highlighted by the long European experience. In fact, the exclusion of computer programmes "as such" from the patentability field has taken the EPO on a crusade of trying to draw up the boundary between patentable and non-patentable objects in software. As already mentioned, the interpretation of the "as such" exclusion revolves around the "technical" criterion. That is: to be patentable subject-matter, inventions must be technical in nature. The difficulty of applying such a doctrine in software, however, has brought both the EPO and some national courts to embrace many different and inconsistent approaches over the years – leading to a general lack of legal certainty in the field.[91] This has also recently led to a referral to the Enlarged Board of Appeal in order to shed some light on the matter.[92] The same inconsistency also seeps out from the American jurisprudence (that precedes the full extension of patent protection to software and business methods inventions under the *State Street* doctrine[93]) and, more recently, from the difficult application of the rather clumsy "machine or transformation" test (from the Bilski case).[94]

An interesting, attempted, solution in the context of patentable subject-matter had also been put forward by the British courts, following the *Aerotel/Macrossan* decision[95] in 2006. This approach was based on the application of a test composed by four steps: first, to properly construe the claim; second, to identify the actual contribution in light of the prior art; third, to ask whether the actual contribution falls solely within the excluded subject-matters; and fourth, to check whether the actual or alleged contribution is technical in nature.

The application of the four-step test has certainly proven successful in reducing the number of CII patents issued in the UK.[96] From this perspective, therefore, the *Macrossan* approach could represent a great instrument for smoothing the patent thicket. Such a practice, however, might not be appropriate from a legal point of view. In Particular, the fact that the *Macrossan* approach goes back to the "contribution approach" that originated at the EPO from the *Vicom* decision[97] is a major point of

[91] See note 82.

[92] See Referral under 112(1)b) EPC by the President of the EPO (Patentability of programs for computers) to the Enlarged Board of Appeal, pending under Ref. N° G3/08 (23 Oct 2008).

[93] *Diamond v Diehr*, note 29 above; *Parker v Flook*, 437 U.S. 584 (1978); *Gottschalk v Benson*, 409 U.S. 63, 67 (1972).

[94] For more details see K Collins, "An Initial Comment on In Re Bilski: Tangibility Gone Meta" (2008). Available at: http://www.patentlyo.com/patent/law/collinsmetabilski.pdf (accessed 25 Feb 2009).

[95] English Court of Appeal, *Aerotel Ltd. v Telco and on the matter of patent application* GB 0314464.9, [2006] EWCA Civ 1371.

[96] See D Bainbridge, "Court of Appeal Parts Company with the EPO on Software Patents" (2007) 23 *Computer Law and Security Report* 2, at 199-204.

[97] See note 30.

controversy.[98] In fact, under this approach the examiner should decide whether there is an inventive step in order to ascertain the existence of an invention. However, under the EPC rules, access to patentability first requires an invention, and only then do novelty, inventive step, and industrial applicability need to be checked. This being so, the legitimacy of the contribution and, thus, also of the *Macrossan* approach, seems somehow lacking.

None of the attempted solutions mentioned have managed to provide a satisfactory answer to the software patent thicket issue which, instead, is still deeply entrenched into the system. What is certain, however, is that the transaction costs involved with the continuous formulation of proposals for new ways of tackling the problem do not only increasingly shift resources from innovation to litigation, but also undermine the purposes for which legal incentives are initially provided.

On the basis of this, this article suggests that the problem related to the software patent thicket should be addressed from a different angle. Specifically, the paper argues that patent law already possesses the appropriate instrument to address the issue, namely the requirement of sufficient disclosure in Europe and the enablement requirement in the United States.[99] A reform to improve the notice quality of issued claims should, thus, primarily come from the Patent Offices. Accordingly, examiners should ask for more information about the meaning of the claims, while rejecting vague and abstract claims more aggressively.

A more extensive disclosure could be a very effective tool to inhibit the growth of the software patent thicket, because it would reduce one of its major causes, i.e. the abstract nature of patent claims. Furthermore, when combined with a good prior art repository, this would support patent officers in conducting a more efficient quality examination, also in terms of inventiveness.

A comprehensive solution should also look at other fronts. For instance, the use of software patents in a more "ethical" way should be sought by the courts through the regulation of the use of injunctive relief. However, since the goal of the present article is to analyse the origin of the software patent thicket, and to propose solutions accordingly, these subsequently-arisen aspects of the problem are left for further investigation.

4.2 The Way Forward

4.2.1 A Better Prior Art Repository

As explained before, one of the major reasons behind the overlapping of IP rights in software is the absence of good prior art repositories. This has led to many non-novel and obvious patents being issued. Various initiatives have taken place recently on both sides of the Atlantic in this respect.

---

[98] For more details see note 82 above. See also W Cook, and G Lees, "Test Clarified for UK Software and Business Method Patents: But What About the EPO?" (2007) 29 *European Intellectual Property Review* 3, at 115-118.

[99] It should be noted that the recent Federal Circuit case law addressing the issue has focused in particular on the written description doctrine. See, for instance, *LizardTech, Inc. v Earth Resource Mapping, Inc.*, 424 F.3d 1336 (Fed. Cir. 2005). See also USPTO Written Description Training Material (25 Mar 2008).

The USPTO Strategic Plan 2007-2012 brought about the emergence of different programmes, mostly led by the open source community and aiming at improving the prior art repository in the field of software-related patents, in an effort to increase collaboration between the patent office, the scientific community and the industry. For instance, the Patent Commons,[100] the Peer to Patent,[101] the Open Source As Prior Art (OSAPA)[102] are some of the most well-known projects. In Europe some proposals have also been put forward, although on a more limited scale. The PatExpert project[103] is one of such examples.

These are all interesting and highly welcome efforts that should be encouraged and that will hopefully lead to more clarity and certainty in the assessment of the novelty and inventive step requirements of software-related inventions. However, it appears clear, not only that much more work needs to be done in this direction (in this respect, both private and governmental projects to clear the registries should be supported) but also that, even though these projects might smooth overlapping problems, they will most likely not entirely solve them. In fact, although searching problems might arise due to the complexity and vast number of patents in the field, absent other additional circumstances, a radical increase in resources and efforts to improve the prior art would, in the long run, solve the matter. In the context of software, however, it is much more complicated.

As extensively explained, the major reason for the growth of overlapping problems is the abstract nature of software patent claims, which, on the one hand, makes it easier for patent drafters to "play around" patent applications; yet on the other hand, it poses big challenges for the drawing-up of the novelty and inventiveness boundaries with respect to the prior art. This is mainly caused by the fact that in software patents the disclosure requirement is particularly weak.

Efforts should thus be addressed, not only at improving the prior art repositories, but also, and most importantly, at reducing abstraction concerns. To this end, the disclosure requirement should be enhanced.

4.2.2 More Disclosure Is Needed

The complexity of contemporary software inventions has led programmers to increasingly rely on high-level programming languages, hiding low-level details. As a result, software engineering has evolved towards higher levels of abstraction, jumping

[100] Patent Common project: http://www.patentcommons.org/ (accessed 25 Feb 2009).

[101] Open Source As Prior Art (OSAPA) project: https://www.linux-foundation.org/en/Osapa (accessed 25 Feb 2009).

[102] Prior aRT and Software Patents project: http://wiki.mozilla.org/Legal:Prior_Art#Summary (accessed 25 Feb 2009). Additionally, important repositories for open source projects in the field are Sourceforge: http://sourceforge.net (accessed 25 Feb 2009) and Freshmeat: http://freshmeat.net/about (accessed 25 Feb 2009). See also Open-IP: http://www.open-ip.org (accessed 25 Feb 2009).

[103] PatExpert: http://www.patexpert.org (accessed 25 Feb 2009).

from simple routines and procedures to more organised units like modules, packages, object classes and components.[104]

Instead of following such an evolution, however, the level of disclosure required in patent applications has remained unchanged, resulting in a generally inadequate instrument to unveil software-related inventions in patents. This is particularly detrimental for the software sector due to the fact that the construction of the claims tends to be much more ambiguous in such abstract technologies and, thus, more extensive information and examples are usually necessary in order to understand and reproduce the invention.

It remains clear that different types of software inventions call for different levels of disclosure. Hence, the optimal level of disclosure in a software patent can vary from a high-level architectural description, to flowcharts,[105] pseudocodes,[106] or source codes.[107] In order to increase patent notice and, in turn, reduce abstraction in software-related patents, none of the mentioned forms of expression should be exclusive. Instead, they should complement each other. This being so, however, at least flowcharts or pseudocodes should always be required in the applications, along with the description of the invention in natural language. In fact, although, in general, flowcharts might be a more workable tool, it is not difficult to imagine situations where flowcharts or diagrams would not be sufficient to show that the applicant is in actual possession of the claimed invention. For example, the path along an arrow from A to B might actually be impossible to practice for a person skilled in the art without the associated pseudocode.[108]

Additionally, when necessary, the source code might also be requested. In these cases the submission in a format like CD-ROM, or other mass storage devices that may be used by examiners to inspect efficiently and effectively the computer programme code listings, would be more suitable than submission in paper. Specifically, as source codes could be considered the "genes" of software, a disclosure policy could be built on the lines of the escrow of biological samples in biotechnology patents. In the same way as for biotech patents, providing the source codes might be required when the invention cannot be described in a patent application in such a manner as to enable it to be reproduced by a person skilled in the art. [109] Moreover, in the same way as for

[104] B Hall, "On Copyright and Patent Protection for Software and Databases: A Tale of Two Worlds", in O Granstrand (eds) *Economics, Law and Intellectual Property. Seeking Strategies for Research and Teaching in a Developing Field*, Ch 11 (Kluwer Academic Publishers 2004).

[105] A flow chart (or flow diagram, or flow sheet) is a schematic representation of a sequence of operations, as in a manufacturing process, or computer programme.

[106] Pseudocode is a form of 'structured English' that looks like the source code of a high level programming language, but is more generic and somewhat more easily readable. Pseudocodes are easier for humans to understand than conventional programming language code, as they typically omit details that are not essential for the human understanding of the algorithm.

[107] Source code refers to any sequence of statements of declarations written in some human-readable computer programming language. Source code allows the programmer to communicate with the computer using a reserved number of instructions.

[108] See note 104.

[109] See Implementing Regulations to the Convention on the Grant of European Patents, Part II-5, Rule 31-32; Directive 98/44/EC of the European Parliament and the Council of 6 July 1998 on the legal protection of biotechnological inventions, Article 13; 37 Code of Federal Regulations (CFR), s 1.808.

biotech patents, access to the deposited material (i.e. the source code) should be restricted until the issuing of the patent, as well as in cases of refusal or withdrawal, in order for the applicant not to lose important trade secrets.[110]

As already mentioned, this policy would find justification in the sufficiency of disclosure (in Europe) and the enablement requirement (in the US) of patent law.[111]

4.2.3 Enhancing Disclosure To Smooth The Thicket

There is reason to believe that requiring software patent applicants to provide flowcharts, pseudocodes or, when necessary, source codes, in addition to the clear description of the invention in natural language, would help in affording a more adequate, less broad, level of protection by reducing one of the major causes of the overlapping problems in software-related patents – that is, abstraction. The enhanced notice would limit the number of patents filed, which would thus also indirectly smooth the thicket.

Generally speaking, the patent system is structured to encourage patent filing early in an invention's development.[112] When applying for a patent in the software field, for instance, one does not necessarily need to have produced any flowchart, pseudocode, nor line of source code. Requiring applicants to provide this information and, in so doing, pushing them to file at a later stage in their idea's conceptualisation, would lead both to less abstract applications and to a reduced number of patents filed. Flowcharts, pseudocodes, or source codes would constitute an additional, better "proof" that the applicant has an invention and, in this way, would slow down the filing of very uncertain applications. Furthermore, it would also make it easier to tighten patent protection to the specific function produced by the mentioned design documents or lines of code. In turn, this would lessen the abstract nature of the claims.[113]

Postponing the applicants' filing time would be possible in the software field in particular because the R&D costs that inventors have to bear in order to create further innovation (thus, the monetary incentives needed) are relatively low.[114] The cost of R&D in software has been made even smaller by current technology that adopts automated tools, for example, to generate sections of codes so as to help designing simple programmes like websites. Moreover, when products are intangible information, as in the case of software, the cost and speed of imitation (as well as the marginal costs of production) is also usually low. In fact, while writing a programme takes relatively little time and money, the debugging of such a programme is still a

See also Budapest Treaty on the International Recognition of the Deposit of Microorganisms for the Purposes of Patent Procedure, 28 Apr 1977.

[110] *Ibid.*

[111] See EPC, Article 83 and 35 U.S.C. § 112.

[112] See J Duffy, "Rethinking the Prospect Theory of Patents" (2004) 71 *University of Chicago Law Review* 2, at 439-510.

[113] See K Rowe, "Why Pay for What's Free?: Minimizing the Patent Threat to Free and Open Source Software" (2008) 7 *John Marshall Review of Intellectual Property Law* 3, at 595-620.

[114] D Burk, and M Lemley, "Policy Levers in Patent Law" (2003) 89 *Virginia Law Review* 7, at 1575-1696.

significant undertaking.[115] With software reverse engineering being an inefficient tool for revealing patented software-related inventions and the source code often being kept secret, copying is usually quite difficult.

Various studies on innovation, in fact, have documented the weakness of using patents for protecting software-related inventions.[116] Companies mostly acquire software patents not to protect their inventions, but rather for "strategic" reasons, such as to support complex cross-licensing agreements, or in response to lawsuits alleging infringement, or as "patent signals" to show to their potential partners that they have a powerful research and development arm, or, again, to increase the company's value.[117]

Overall, this analysis suggests that increased disclosure would not reduce innovation incentives in software. On the contrary, it would greatly benefit both the patent examiners and the third parties in understanding the patented inventions and, in so doing, it would provide more incentives for further progress. Major reasons of concern, in fact, are that software-related inventions do not transpire "from the face" of the patents – they are very obscure and not readily understandable and, consequently, difficult to be built upon. This is also one of the justifications of why software-related patents are mostly ignored by researchers and companies.[118]

Reducing the abstract nature of the claims would also help to improve analysing applications in terms of prior art. As seen, the abstract nature of the software patent claims makes it difficult for patent examiners to draw the boundary between allegedly new, inventive inventions and prior art: comparing abstract concepts so as to find relevant pieces of prior art is a remarkably challenging task. Lowering the level of abstraction would, to a certain extent, also reduce this problem.

Although this type of policy would mostly address future applications, a better disclosure would also provide great support for the earlier mentioned projects[119] that aim at improving the prior art repositories and clearing the "bad" patents issued.

Another advantage, deriving from a flowchart-code disclosure policy, relates to the breadth of the patent scope. In incremental industries, as is the case with software, in order to maximise progress, the patent scope should be kept limited and patents should not extend to several product generations: this would, instead, stifle innovation in subsequent incremental developments – representing a huge hindrance to

[115] *Ibid*.

[116] *Ibid*. See also W Cohen, R Nelson, and J Walsh, "Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufactory Firms Patent (or Not)" (2000). NBER Working Paper Series No 7552, available at: http://www.nber.org/papers/w7552 (accessed 25 Feb 2009); A Arundel, "The Relative Effectiveness of Patents and Secrecy for Appropriation," (2001) 30 *Research Policy* 4, at 611-624.

[117] See, for instance, C Long, "Patent Signals" (2002) 69 *University of Chicago Law Review* 2, at 625-680. See also R Merges, "Software and Patent Scope: A Report From The Middle Innings" (2007) 85 *Texas Law Review* 7, at 1627-1676.

[118] See M Lemley, "Ignoring Patents" (3 Jul 2007). Stanford Public Law Working Paper No. 999961. Available at SSRN: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=999961 (accessed 25 Feb 2009). See also R Mann, "Do Patents Facilitate Financing in the Software Industry?" (2005) 83 *Texas Law Review* 4, 961-1030.

[119] See notes 100-103.

technological progress.[120] Increasing disclosure would also help in meeting these goals: as the scope of protection in patent law is defined by the claims,[121] requiring applicants to provide more information on their inventions would directly result in a reduction on the scope of the patents as the enhanced notice would limit the claims.

Additionally, providing greater information would support judges dealing with infringement cases in delimiting the scope of protection of the challenged patents. In fact, although it might be easy for someone skilled in computing to understand whether there is the same invention (i.e. the same function) behind two allegedly different software-related patents (by using, for example, reverse engineering techniques) it may be unclear in the eyes of non-technically trained judges. Thus, more detailed information, as well as different tools, might be necessary in order to convince a court that certain software is or is not infringing upon a patented one.

It is important to stress that this article does not suggest requiring applicants to file only once a complete product has been developed (e.g. when the entire source code is available). Although such a condition would most likely fulfil the disclosure requirement in the majority of cases, it would clearly discriminate patentees in the software arena due to the great deal of work experimentation that would be necessary before the filing of an application. In this way, a mandatory source code requirement would constitute a degree of disclosure that would go far beyond the norm for the other kinds of patents, and could thus raise legitimacy concerns under Art. 27 and 30 of the TRIPs Agreement,[122] which prohibits EU member states from discriminating on the granting of patents based on the type of technology at issue.[123] A full source code disclosure, in fact, would require that an invention be disclosed so that, not only a skilled programmer, but also a person of virtually no programming experience would be able to make it and use it. Such a requisite would also most likely be highly inefficient from a practical point of view: providing too long a description in a particular programming language could, in fact, make the disclosure even worse – drowning the important facts into the unimportant ones. Additionally, it could create searching problems, delaying filing, as well as the examining and issuing of patents in the field. Overall, this could degenerate into either too broad or too narrow patents being issued.

What this article proposes, instead, is to improve disclosure functions by affording more power to examiners that allows them to reject very abstract and unclear applications, and, especially, by pushing applicants to always use flowcharts or pseudocodes along with a clear description of the invention in natural language. Computer programme code listings, instead, should be requested only in the case and to the extent that they are necessary to enable the skilled man. This would burden applicants or examiners in the software field no more than in any other field of technology.

---

[120] See note 109.

[121] EPC, Article 69; 35 U.S.C. § 112.

[122] WIPO Treaty on Trade Related Aspects of Intellectual Property Rights, Morocco, 15 Apr 1994 (TRIPs).

[123] See G Dinwoodie, and R Dreyfuss, "Diversifying Without Discriminating; Complying with the Mandates of the TRIPs Agreement" (2007) 13 *Michigan Telecommunications and Technology Law Review* 2, at 445-456.

4.2.4 Possible Drawbacks: Some Remarks On Trade Secrecy

Although increased notice in software patents applications would bring important advantages, it could also potentially raise some concerns. For instance, it could be argued that enhanced patent disclosure could lead to a more extensive use of trade secret protection. Overall, trade secrets might be even more detrimental than patents because information is kept undisclosed. Some characteristics of the software industry, however, suggest that such a threat is probably not very likely.

In the early days of software protection the use of trade secrets was thought to be problematic in the code context because of the ease with which the secret status could be lost.[124] At a time like the one in which we are living – where DRM mechanisms are becoming increasingly sophisticated and, in particular, where software developers usually do not disseminate source code without first encrypting it, using passwords, or restricting access to information placed on computers – trade secrecy has certainly become a more useful and appropriate instrument of protection for the software code. Nowadays, in fact, inventors simultaneously combine the use of trade secrets and patents, as well as licensing agreements and DRM to protect the functional aspects of their software.

For widely distributed commercial software, however, keeping information of the programme completely secret is a highly challenging task. This is made particularly difficult by the fact that outsourcing work overseas has become increasingly common.[125] Additionally, when the trade secrets are maintained through licencing contracts distributed with the software, enforcement is almost impractical as these contracts are constantly ignored. Trade secrecy might also lead to very inefficient practices, like keeping the entire code undisclosed. In component industries like computer software, though, the need to access sections of the codes so as to "fit the pieces together" might often be necessary.[126]

Finally, even if enhanced patent notice would lead to less information being disclosed, it is not immediately clear whether this would hurt innovation in the software field in the long run. Recent studies have demonstrated that researchers and companies in component industries mostly ignore patents.[127] Under current rules, in fact, software patents are so obscure that they are effectively secrets. This failure of software patents to pursue their traditional function of spreading technological knowledge indicates that most innovations in the field are "independent" inventions.

This not only supports the thesis that disclosure does not work particularly well in the software context but it also suggests that a potential increase in the use of secrecy would not reduce innovation in the field. On the contrary, if anything, it would probably curtail the dense web of patents by discouraging the filing of too abstract and uncertain applications.

---

[124] K Canfield, "The Disclosure of Source Code in Software Patents: Should Software Patents Be Open Source?" (2006) 7 *Columbia Science and Technology Law Review* 6.

[125] *Ibid.*

[126] *Ibid.*

[127] See note 114.

5. Conclusions

The problems related to software patent thickets have been openly recognised by scholars, courts and legislators. Accordingly, in recent years, various initiatives have been put forward to address the issue. Efforts, however, have thus far been focused merely in improving prior art repositories; curtailing the patentable subject-matters; increasing the non-obviousness bar; and reducing the use of injunctive relief. Issues related to the abstract nature of software have, on the contrary, not been properly assessed. However, this article demonstrates that abstraction is one of the major causes of the growing software patent thicket and, as such, represents a fundamental issue to be addressed.

Specifically, this article suggested that a more extensive disclosure, when combined with a good prior art repository, can be a potentially successful tool for reducing the effects of the abstract nature of software-related patent claims and, in turn, inhibit the patent thicket. To this end, inventors should be requested to include in their applications the flowcharts, pseudocodes or, when necessary, parts of the source codes along with a clear description of the invention in natural language.

Overall, this would not only serve patent officers in better evaluating the applications, but would also support judges dealing with questions of infringement. Competitors and new inventors – who necessarily need detailed information about the programmes behind the patented inventions in order to improve upon them – would also clearly benefit from such a practice.