3-2012

Software Protection: Copyrightability vs Patentability?

Deli Yang
Trinity University, dyang@trinity.edu

Journal of Intellectual Property Rights
Vol 17, March 2012, pp 160-164

Global IP Debates

Software Protection: Copyrightability vs Patentability?

Deli Yang†
Department of Business Administration, Trinity University, One Trinity Place, San Antonio, TX 78212-7200, USA

Received 16 February 2012

The moment software was created was also the starting point of heated debates over software protection. During the early stages of protection, the debate was around the copyrightability of software protection. Nowadays, the focal point is the argument on whether software should be solely protected under copyright or dually guarded by both copyright and patent. With the development of the software industry across the world, this issue has become more and more contentious internationally. This column of *Global IP Debate* opens with a software patent case and an open source case, and then traces the history of software protection to examine its protective evolvement. It next focuses on the currently heated debate of dual protection on software by laying out both sides of the arguments and reasoning. This debate ends with some possible solutions to engage readers' thoughts.

Keywords: Software, software protection, software copyright, software patent

Landmark Cases in Software Protection

Software is defined as a 'computer programme' with a set of statements or instructions used in a computer to bring about a certain result.[1] It is detailed as 'computer-implemented invention...an expression intended to cover claims which involve computers, computer networks or other conventional programmable apparatus whereby *prima facie* the novel features of the claimed invention are realized by means of a programme.'[2] Two relevant cases symbolize the beginning of software protection, but in different ways.

The *Diamond* v *Diehr* case in 1981 marked the basis for software patents in the world.[3] The court reviewed the case of patent application, essentially a computer-aided process using mathematical formulas to operate a rubber-curing machine (time, temperature and thickness of the mould) to produce cured precision products from raw synthetic rubber, that was rejected at the US Patent and Trademark Office (USPTO). The patent examiner concluded it to be unpatentable subject matter, and the board appeal in the USPTO affirmed the decision, but the Court of Appeals for the Federal Circuit reversed the decision by emphasizing that a patentable invention did not become unpatentable simply due to a computer being involved. Subsequently, the US Supreme Court upheld the early decision that a machine or process using a mathematical algorithm, including a software component, which is different from mathematical formulas in the abstract, was patent-eligible as a whole. Accordingly, a patent certificate (No 4344142) was issued entitled 'Direct digital control of rubber molding presses' with 11 method claims. Since this court verdict, it has been clear that business methods and software-related inventions are patentable in the US, when properly claimed and if they do not preempt an abstract idea. However, the subsequent burden lies on the USPTO and lower courts to decide the distinction between the two when examining a related application.

The other opening case, *Jacobsen* v *Katzer*,[4] concerns recognition of protection and enforcement of open source software from year 2006. DecoderPro, an open source collaborative project allowed hobbyists to share open source software to control model trains. Jacobsen, as the manager of this collaborative project placed the DecoderPro software in the public domain for sharing on the condition of abiding by the copyleft licence agreement. That is, anyone could use, modify, and distribute the software on condition of publishing the software improvement and making it available to the public. By indicating the detailed modification of the source code, location and time to do so, contributions by different programmers could be identified and recognized. Matthew Katzer's Kamind Associates Inc built on the software and developed competing commercial software. Although portions of DecoderPro were used, the company did not comply with the copyleft licence terms.

†Email: dyang@trinity.edu

Jacobsen sued Katzer for copyright infringement and requested a preliminary injunction, but Katzer contended the licence was non-exclusive and public, thus there was no copyright infringement. The District Court in California denied Jacobsen's motion on the ground that the copyleft terms did not have a clear scope and the terms between these two parties were only binding under contract law. Jacobsen subsequently appealed to the Federal Circuit. In 2009, the special court ruled that the copyleft conditions set out by the Jacobson collaborative were enforceable to prevent copyright infringement. Eventually, the two firms settled their dispute outside the court.

This case confirms the legitimate right of open source software to control any modification and distribution, and gain potential economic benefit. This was the first time an appellate court handled an open source case, its relevant legitimate right, and appropriate remedies. This case also clarifies a few misconceptions. First, open source software is not 'free' software. Although it is available in the public domain, preconditions exist for using the software. Second, it is important to abide by the preconditions; otherwise, it will cost the ignorant user money and reputation.

The two cases clearly indicate that regardless of whether it is open source software or proprietary software, owners can seek protection and enforcement of their right under copyright and/or patent right. However, things are not as straight forward as they seem, particularly, when software protection involves cross-border activities. Scholars and practitioners have been constantly debating on whether software should be protected under copyright, or given dual protection of copyright and patent. Under such circumstances, stipulators in different countries appear to have adopted a 'wait and see' approach to handle the contentious issue.

Concept and Brief History of Software Protection

Computer programmes and their protection tend to focus on three elements: object code, source code and documentation. Source code is the original code in programme languages, and can be read by specialists in the field. Object code is the code read by a computer and humans can only read it when transformed into source code. Firms use their own established technical systems to protect their software, such as encryption, firewalls and passwords. They also rely on legal means to protect their ownership right. Under the current legal environment, firms can protect their software under the laws of trademark, copyright, patent or licensing contract, depending on the country concerned. For example, proprietary software firms rely on copyrights to protect their ownership, prevent any unauthorized copying, and defend against the violation of their legal right. Open source software also depend on copyright protection to ensure enforcement of their copyleft licensing terms. (i.e. licensees are obliged to share their modifications and improvement of the software work). The US and Japan also grant patent protection of software to prevent others from using, making or selling the patented software without authorization. However, software protection has always been an issue of national and international debate from its early history under copyright protection to the current on-going discussion under patent protection. One reason for such a debate is that many do not view software as a tangible object, or object code as a creative work.

Nowadays, copyright protection of software is widely accepted across the world, but the early history of copyright protection for computer programme took a long time to evolve.[5] Since the 1960s, software gradually emerged as a dynamic industry. Before then, software was not used for commercial purposes, but in-house programmes of the computer firm. Protection became a central concern since software became a commercial object for distribution. The Copyright Act in the US at that time (1909) required copyright registration upon the first publication of any work. The Copyright Office received its first deposit, submitted as a tape, of a computer programme in 1961, and subsequently a Columbia University law student submitted two computer programmes (a printout and a magnet tape). All these three programmes were registered for copyright protection in 1964 with the understanding that they were not ordinary books, but 'how to' books. When the Copyright Act of 1976 was enacted (effective in 1978), software was officially accepted for copyright protection. Despite the official recognition of copyright protection for software, it was still unclear how software should be protected as a unique copyrighted work. For this reason, the National Commission on New Technological Uses of Copyrighted works (CONTU) was appointed to make further recommendations about how to protect software under the Copyright Act. In this context, CONTU not only defined software, but also clarified the owner's right to make and adapt their software for use.

Following the US, there were efforts from other developed countries toward software protection in the 1980s; meanwhile, WIPO and UNESCO convened an expert meeting on copyright protection of computer programmes in 1985. This meeting resulted in speedy recognition of copyright protection for computer software across the world. Now, the TRIPS Agreement also clearly states that both source and object codes of computer programs be protected by the Berne Convention.

Unlike software protection under copyright, software was not patentable prior to the 1970s. However, the *Diamond* v *Diehr* case changed the software patent history. As a consequence, the Federal Circuit was burdened with the responsibility to clarify patentability of software. Thus, although invention as a mathematical algorithm is unpatentable, an invention using a computer programme to enhance operational productivity is patentable.

With the US setting the precedence by allowing software patenting, countries like Japan and a few European countries also followed suit and permitted patent protection of software. For example, the UK does not allow stand-alone software patents, but permits functional software invention to be patented under its Patents Act 1977. As a result, heated debates then shifted from the US to Europe, not only among the nations, but also between large and medium-sized software enterprises.[6]

Unlike the wide acceptance of software copyright protection, software patent is not an international statutory requirement set out by TRIPS under the WTO or by other treaties, conventions or agreements under WIPO. Worldwide, Japan and the US allow software patenting, while nations like India explicitly emphasize otherwise.[7] Most countries, however, adopt an 'it-depends' approach. For example, the State IP Office's Guidelines for Examination state that computer programmes are not patentable in China, but they can be considered for patent protection if they solve a technical problem and use a technical measure to generate effective outcome. The EU practice is similar, in allowing grant of software patents only when there is a visible technical contribution. The EU stipulations on software protection has not changed its legal position, rather, the intention was to harmonize the software protection environment[8], as most software patents were granted to American and Japanese owners.[9] Given the nation-based nature of IP protection, to patent or not to patent software is likely to create conflicts of interests for cross-border businesses.

Debates over Software Protection and Dual Protection of Copyright and Patent

Following on the historical account of software protection, this section centres around the key argument: should software be protected under copyright only or accorded dual protection under copyright and patent.

The grant of a patent gives an IP owner the legal monopoly to exploit an invention for 20 years. In this case, therefore, software is treated as an invention, and published work for authorized exploitation so that society can reap benefits. Meanwhile, copyright protection protects artistic and literary expressions. When it comes to protecting software, there is significant controversy as to which structure is most suitable for the programme code at the core of software development.

Proponents of copyright protection argue that it would facilitate the development of software for the following reasons. Firstly, applications for copyright protection do not need to go through a granting process: copyright automatically authorizes ownership to creators. Second, there is less fear of infringements. Each new software programme represents a new copyright, and the creators (software writers in this case) need not worry about whether there is an intrusion of prior art; thus each new programme may contain knowledge originally created by previous designers. Third, licensing deals to commercialize the software are both simpler and cheaper. A potential licensee of a software programme need not approach many owners (of previously copyrighted elements) to arrange for commercialization. This would attract wider interests for disseminating new software knowledge at much lower costs, thus stimulating both competition and industry development. Proponents of copyrighting as the indicated protection model for software argue that 'the main use of software patents is to block out competition.'[10]

Like advocates for conventional patents, supporters of software patenting believe that patent protection provides programmers with incentives for further research and development. Patents authorize owners with a period of monopoly, and stimulate competition by developing new software, thus avoiding licensing fees, and encourage rapid development of software technology and commercialization due to required invention disclosure. Advocates also argue that patents protect functionality that allows only one owner or joint owner for each patent, which in

consequence, also helps to prevent infringement. Functional protection under patent right can not be achieved by copyright protection due to its focus on protection of expressions.

However, software patenting is also widely opposed, and its opponents cite many drawbacks. First, they argue that it would be absurd to patent software because of its inherently different nature in comparison to patentable technologies. A software does not wear out, and compared to a conventional invention, which may only involve a few patents; it may contain 100,000 or even 10 million lines of code, involving perhaps 1,000 patents. Thus, examining patentability, particularly originality and non-obviousness, could be extremely complex issues. Moreover, the timescales involved make patenting unsuitable. While 20-year protection might suit conventional industries, such levels would be pointless for software, where product life-cycles may be five years at a maximum, and (given the time involved in application and examining procedures), it is easily conceivable that the software could be outdated even before the patent is granted.

Second, the argument against software patenting is the dangers of monopoly. Patenting is designed to protect fair competition and encourage cross-licensing activities among firms. But, in reality, software patents are highly concentrated among a few large companies (such as IBM, AT&T, Hitachi, Toshiba, Xerox and Microsoft), who are able to recruit high calibre software engineers to develop programs and have the market power to commercialize software widely. In contrast, SMEs, with limited resources in these areas, are faced with destructive levels of delay and costs when they advance a software programme for patenting , due to the density of the codes involved. Their only other option is to license software from one of the big players, and subject themselves to crippling royalty payments. Either way, the patent system is not ensuring fair competition for the ultimate benefit of the consumer.

Third, software is easy to infringe and this problem impacts both large and small firms. Modern software is a highly sophisticated product, which can involve so many patents that it is very difficult to be sure to what extent independently written new software is truly original. The abstract and broad nature of computer software adds to the search complexity and consequently, unintended infringements occur frequently. A typical case is the dispute between Microsoft and Stac Electronics.[10] In 1993, Stac, a small software house, filed a case against Microsoft for infringing Stac's data compression system in its MS-DOS 6 operating system. Stac claimed that the infringement had meant it had had to lay off 20 per cent of its work force. In the end, the court ruled that Microsoft should pay US$ 120 million, but it was allowed to make a counterclaim of US$ 12.6 million against Stac because it had used Microsoft technology to develop its MS- DOC stacker.

Fourth, software is slow to disseminate under patent protection. Given the sophistication of the product, the length of time involved in navigating existing software patents and the cost of analysing the programmes, together with the fear of litigation, patented software attracts only a limited circle of software development engineers. It is not surprising one of the constant complaints in the software industry is that ideas are plentiful, but that development is slow and inadequate.

Finally, the continued uncertainty about how best to protect software developers' interests causes problems of inconsistency among countries in terms of policy making. A salient example is the European Parliament's dilemma about whether or not to pass the Software Patent Bill in July 2005. Faced with the decision whether to support hi-tech firms' R&D or jeopardize SMEs and open source developers by agreeing to adopt patenting , the parliament rejected the bill by 648 votes against 14, resolving that software should only be copyrighted within the EU. However, this outcome did not resolve the inconsistency among countries, even within the EU, regarding the rules for software protection. Thus, uncertainty continues to exist, producing further confusion about protection levels and anxiety about litigation.

The case against software patenting has gained a large circle of supporters among SMEs, scholars, government officials, and is backed by increasing numbers of corporate and organizational advocates, such as the Association for Computing Machinery, WordPerfect, Borland International, Oracle, AutoDesk, Open Source Initiative (OSI), Free Software Foundation, League for Programming Freedom, Foundation for Free Information Infrastructure and InfoWorld. Software originators, such as Bricklin (spreadsheets) and Kapor (Lotus) are also seriously concerned about the future of software development under the restrictions of patenting.

Patent opposers believe government should provide policy support for software development instead of imposing software patenting. In fact, as Oracle reports, many companies who are not supporters of the argument nevertheless feel forced to patent their software in order to avoid litigation.

Proposing Possible Solutions

Software protection will continue to be a long term debate, but the focal point in the international realm is likely to shift with time from the focus of 'for or against' software patent to the more specific and practical step of harmonization and development given nations' shared destiny: software is part of people's life in every way. Within this ambit, some possible solutions can be considered.

Firstly, given the disparity among nations toward software patenting, WIPO and the WTO should clarify their stand on software patents rather than leaving nations to their own choices. If software patenting were allowed, it would reduce the degree of confusion by defining a specific scope of software patenting. The main benefit of doing so would be to set a broad standard for nations to comply with. This would lead to less conflict across countries when doing international business involving software protection.

Secondly, nations play a key role in resolving potential disputes associated with software protection. Thus, they need to clearly define what can be patented and what cannot be patented if they are not able to 'ban' patent protection. In addition to emphasizing the effect of resolving a technical problem and unpatentability of stand-alone software, as in many countries, nations should have clear stipuations as to what are/are not patentable in terms of software protection, and what specific legal procedures to follow if cross-border disputes occur.

Finally, given the interconnected nature of patent software, owners may establish an open-source-software-like association for proprietary software in which patented software owners under a certified association mark, can bundle relevant software together for further development under cross-licensing agreements. This action will help accerlerate technology spillovers, reduce protection and commercializing costs, and draw the attention of experts for further research and development without concerns about patent infringement. Such an action will also, in the long run, help standardize the software industry.

References

1 *Final Report of the National Commission on New Technological Uses of Copyrighted Works* (National Commission Washington DC), 1978, http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/US_LOC/L780731N.pdf (12 February 2012).

2 *Guidelines for Examination in the European Patent Office* (2007), European Patent Office Munich, http://www.epo.org/service-support/publications/procedure/guidelines-epc2000.html (12 February 2012).

3 *Diamond* v *Diehr*, 450 US 175 (1981).

4 *Jacobsen* v *Katzer*, No C06-01905 JSW, 2006.

5 Hollaar L A, *Copyright of Computer Programs*, 2002, http://digital-law-online.info/lpdi1.0/treatise17.html (10 February 2012).

6 Story Alan, Intellectual property and computer software – A battle of competing use and access visions for countries of the south, UNCTAD, Issue Paper No 10 (2004).

7 Rao D and Gopinatham S K, Argument rages on over software patenting, Section: India: Sponsored editorial, *Managing Intellectual Property* (1 September 2009).

8 Zekos G, Software patenting, *Journal of World Intellectual Property*, 9 (4) (2006) 426-444.

9 Guntersdorfer M, Software patent law: United States and Europe compared, *Duke Law and Technology Review*, 0006 (March 2003).

10 Fisher L M, Microsoft in accord on patent, *The New York Times*, 22 June1994; Riordan T, (1994) Patents; A software-technology infringement case against Microsoft goes to trial in Federal Court, *The New York Times*, 24 January 1994.