ASIPI
XIII Work Sessions
Panel No. 4
Cartagena, Colombia
December 3, 2007

PROTECTION OF SOFTWARE VIA PATENTS

Abstract

An often quoted axiom of patent law is that "anything under the sun that is made by man" is patentable. However, with software patentability is not a foregone conclusion. For a little more than a quarter century, software has been accepted in the U.S. as patentable subject matter. Contentions over how much protection will be afforded to software, how to restrict software as statutory subject matter, and even the financial viability of software patents for industry is still in the balance.

The first part of this paper outlines the development of software patents, the current USPTO regulations, current cases shaping software patent protection and gives an overview of the difference between US, European, Japanese, and Korean software patent protection. The second part of this paper gives an example of software patent enforcement, examines the impact of software patents on innovation and entry into the software market, and briefly outlines how Open Source Software Licenses can affect software patents.

The third part of this paper describes how software patent protection is compatible with other types of Intellectual Property Rights (IPRs) and indeed filled a hole in the regime for protection of software. Specifically, software patents complement copyright and trade secret protection. The final section of this paper postulates that *sui generis* protection would be the most suitable form of protection, as was proposed when the issue first arose in the mid-sixties.

1. Software Patents

a. A Brief History

Prior to 1981 software patents were not accepted by the United States Patent and Trademark Office (USPTO) or the courts as patentable subject matter. The USPTO refused to allow software patents and this view was shared by the United States Supreme Court.[i] In cases such as *Gottschalk v. Benson* decided in 1972, the Court rejected software patents as patenting algorithms or mathematical equations without a specific application.[ii] The Patent Office then successfully rejected a string of software patents until *Diamond v. Diehr*.[iii] This case was appealed to the Supreme Court, where the Court stated that an algorithm for curing rubber was patentable subject matter, although presented in software as an application of an algorithm to a process.[iv] This holding allowed software to be patented in conjunction with a medium of operation.

The reach of software patents was extended with *In re Beauregard*[v]. Set up by IBM as a challenge to current USPTO practice, this case incorporated a software patent into not only a computer but also into a distributable medium.[vi] That is, the patented software was claimed with respect to its storage medium, e.g. on a disk or CD ROM.[vii] While on appeal to the Federal Circuit, the USPTO changed its rules and allowed this type of software claim before any ruling was ever made.[viii]

In re Beauregard was extremely important as it allowed, for the first time, a software algorithm stored on a computer readable medium to be patented. [ix] This allowed the patent holder to directly sue the distributor of infringing software with a software patent. Previously, the patent holder could only sue the end user or the distributor for contributing to patent infringement. Now, instead of suing software distributors under the harder to prove standard of contributory infringement, the distributor could now be named as directly infringing on the software patent.

The scope of software patent protection was further expanded in *State Street Bank & Trust Co. v. Signature Fin. Group* stating that business methods were indeed patentable subject matter.[x] *State Street* stood for the proposition that as long as software produced "a useful, concrete and tangible result," in this case "a final share price momentarily fixed for recording and reporting purposes," that software could be patented.[xi] *State Street* went further in that it also removed any subject matter exception under 35 USC 101 for business method patents. In the wake of *State Street*, Congress passed the inventor's protection act which categorically granted prior user rights to those who had previously used a now patented business method.

Following *State Stree*t, the USPTO began to apply a technical arts tests requiring that applications not be issued to abstract ideas not associated with a "technical art." However, this test was struck down by the Board of Patent Appeals and Interferences (BPAI) in *Ex parte Lundgren*. This application was directed towards a method for incentivizing a manager of a firm. The Examiner rejected this application as not directed toward a "technical art" but only an abstract idea. The BPAI reversed stating that no such test existed. Rather, software must only produce a useful, tangible, and concrete result.

b. Current Patent Regulations

Given this background, the current USPTO guidelines regarding statutory subject matter have been promulgated in the Manual of Patent Examining Procedure (MPEP), section 2106, with sub section 01 devoted to non-statutory computer related subject matter[xii]. Section 2106 outlines specific enumerated unpatentable categories that are not statutory subject matter. These categories include laws of nature, abstract ideas, and pure mathematical expressions and states that inventions must provide a useful, tangible, concrete result. Further, any patentable invention must be a process, machine, or article of manufacture. Section 2106.01, applies section 2106 specifically to computer programs.

Section 2106.01 divides software programs into two distinct classes: functional descriptive material and nonfunctional descriptive material. Functional descriptive material, such as software programs, can be patentable subject matter provided that they are claimed with respect to a computer readable medium. That is, the software itself must be fixed and change or have changed, in a tangible manner, the computer or a storage medium, such as computer memory, a

computer hard drive, or a computer readable medium such as a CD. Section 2106.01 states that non-functional descriptive material, such as music, literary works, compilations, or mere arrangements of data, are not patentable, even if combined with a computer readable medium. Under US law, these types of descriptive matter would have to be protected under a different intellectual property right, such as, copyright protection. Further, contrasting European law, database or data structures are considered to be non-functional descriptive material and are therefore unprotectable under US patent law.

Of course, applications for software patents must also comply with 35 United States Code (USC) sections 101, 102, and 103. Section 101 states that the software must be novel and useful. Section 102 states that the software invention must be novel and defines statutory bars that limit patentability. Section 103 states that the software must also be non-obvious in light of the current knowledge of one of ordinary skill in the art.

c. Current Cases Challenging Patentable Subject Matter

Software protection in the US is still in a state of flux and the aforementioned USPTO rules on software are continually changing as the result of adjudication. Numerous appeals from USPTO patent application rejections are before the Court of Appeals for the Federal Circuit (CAFC), the de facto patent Court in the US. These will shed light on whether a computer is necessary to support a finding that the software creates a useful, tangible, and concrete result. One case will hopefully resolve whether or not signal claims are patentable statutory subject matter.

In re Bilski claimed a method of optimizing management of risk or a method of doing business. In *Bilski's* application, the method was not tied to a physical structure, nor did the claims provide for any transformation of matter. Rather the method could be performed by humans without even "implicit transformation of electric signals from one state to another." *Bilski's* application was interpreted by the BPIA, in a non-precedental opinion, to be nothing more than an abstract idea not producing a tangible, concrete and useful result and therefore affirmed the rejection in this case as non-statutory subject mater under 35 USC 101. This case has been appealed to the Court of Appeals for the Federal Circuit (CAFC) and is currently pending.

In re Cominskey, decided by the CAFC in September 2007, involved a method of placing an arbitration requirement into a document; a method that does not need a machine to be executed. The CAFC decided, *sua sponte* in oral arguments, to request additional briefs inquiring into whether or not this method was statutory subject matter under 35 USC 101 and ultimately rejected the claims as unpatentable under 35 USC 101. The court stated that some of the claims "seek to patent the use of human intelligence in and of itself." The court reiterated earlier holdings that an algorithm or abstract idea is statutory subject matter only if it operates on, transforms, or otherwise involves one of the statutory subject matter classes of 35 USC 101. It therefore held unpatentable under 35 USC 101 the claims which were divorced from statutory subject matter and remanded the other claims, containing statutory subject matter for patentability determinations by the USPTO under 35 USC 103.

Ex parte Nuijten, decided by the CAFC in September 2007, involved a challenge as to whether or not signals, in isolation, are patentable. In this case, the CAFC decided that Nuijten's "transitory, propagating signals" are not patentable as the signal claims do not fall within any of

the four enunciated types of patentable subject matter: processes, machines, articles of manufacture, and compositions of matter. This is because the claim language, as written, was "not limited to by any specific physical medium nor did the dependant claims add any physical limitations." Rather, the CAFC determined that Nuitjen's signal had no physical attributes and therefore described abstract characteristics and unpatenable under *Diamond v. Diehr*. The CAFC clarified their decision in *State Street* stating that these four categories are indeed relevant in determining patentable subject matter, although there is little importance in which specific category the invention may fall. *En banc* review has been requested.

d. Software Patent Protection in the U.S. Compared to that of Europe

Software protection in Europe is different than that in the US. Intellectual property rights for software in Europe are broadly governed by the Trade-Related Aspects of Intellectual Property Rights (TRIPS) agreement, which creates a set of minimal patent standards. TRIPS further requires compliance with the Paris convention, the Berne convention, and the Rome convention. While TRIPS does not list excludible subject matter it does allow member countries to exclude certain statutory items. Therefore, under TRIPS, Europe must grant copyright protection to software, but Europe's software patent protection is narrower than US law.

The European Patent Convention (EPC) is the governing law of patent protection within Europe, although each European country maintains its own patent laws and patent office. EPC §52(2) lists items that are not considered to be patentable, among which is software "as such." The phrase "as such" has been interpreted to only exclude patent protection for computer programs. When software becomes part of an invention or influences the manner of working of a device or controls it and a further technical effect is present, then the software may be patentable. This "further technical effect" requires more than simply modifying the electromagnetic signals within a computer.

The Board of Appeals has stated that this "further technical effect" could result from the software solving a technical problem. An example of this would be software integrated with a computer, the software controlling an industrial process or the working of a piece of machinery. Another example would be software as a means for a specific technical effect created internal to the function of the computer itself; when software is loaded into a computer and provides a contribution to the art it may be patented.

The requirement of "further technical effect" is in contrast with that of the U.S. patent law. The USPTO will allow software, as such, to be patentable provided it is incorporated with a computer readable medium and is new and novel under 35 USC 102 and 35 USC 103. That is, the test the U.S. currently applies is that the software produce a useful, concrete, and tangible result. These differing standards have resulted in much confusion as to the extent of software protection available in Europe.

e. Software Protection in Japan and Korea

Software protection in Japan appears to be fairly similar to that of US patent law. Per Chapter 1 of Part VII of Japanese Examination Guidelines for Patent and Utility Models in Japan, software is patentable if it is a statutory invention where the claimed invention is "a creation of technical ideas utilizing a law of nature"; where "information processing by software is concretely realized

by using hardware resources" is said to be such an utilization of a law of nature. Japanese patent law also allows patenting of program claims, which differs from U.S. patent law in that the software can be separated from the physical medium. However, the Japanese patent law 36(6)ii restricts out "program signal" and "data signal" claims as not being directed to statutory subject matter. 36(6)ii also places restrictions on what type of program product claims can be allowed.

In Korea, according to a newsletter from Bae, Kim & Lee IP Group, the definition of protecteable inventions had been changed to include computer programs.[xiii] Specifically, the definitions in Article 2 of the Patent Act in Korea were changed to include computer programs and transmission through information network. This means that computer programs, which were previously not protected, will now be included in the Patent Act. As well, the changes in these definitions also allow protection of CD-ROMS, disks, and transmission of software through a network, such as the internet.

2. Software Patents in Action: Litigation, Innovation, and OSS

a. Software Patents Enforceable in Litigation,

Software patents are viable ways to protect software and are enforceable in litigation. In the now vacated *Eolas v. Microsoft* case, Eolas was awarded $524 million dollars because Microsoft infringed on Eolas' software patent.[xvii] In another case, ePlus Inc., of Herndon, Va., was awarded a $37 million settlement in a patent infringement suit against rival Ariba Inc.[xviii] Such awards have also resulted in defensive patent filings.

These defensive patents can be extremely valuable to a company, not only to dissuade other companies not to sue, but also to protect a counterclaim if litigation is initiated. An example of defensive patent use is by EMC Corp. against HP Computers.[xix] In this case, HP sued EMC Corp. for patent infringement and EMC counterclaimed with several patents of its own portfolio, upon which some seminal HP technology infringed.[xx] This resulted in a 325 million dollar award, the largest defensive patent award, for EMC against HP.[xxi] Software patent litigation does not show any signs of going away:

> "The number of patent lawsuits continues to rise and so does the size of settlements and judgments, says the Coalition for Patent Fairness, a group supported by large tech companies. Before 1990, only one patent damage award larger than $100 million had been awarded; in the past five years there have been at least 10 judgments and settlements of that size and at least four that topped $500 million, the group says."[xxii]

b. Software Patent Fears Unrealized: Increased Revenue, and Continued Innovation

Opponents of software patents stated that software patents would limit innovation and entrench the established software companies. They feared the threat of litigation would shut out new businesses, that software patents were a bad policy, and that the patent office has issued too many poor software patents.[xxiii] They stated these issued software patents would create a minefield, making it harder for companies to navigate in the software arena.

However, empirically, their fears have not been realized. Software revenue has continued to increase at a healthy clip of 16.5% in 2005 and 14.1% in 2004. New software financing has, in

general, increased from around 500 million dollars in 1995 to 1.8 billion dollars in 2005. New companies continue to enter the software market. Countries seen as incubators of innovation, such as Israel and the US, continue to file more and more software patents. Established industry players, such as AT&T and Xerox, who spend heavily on acquiring patents, have not prevented new firms from entering the market or slowed innovation in the software field.

An explanation offered is that software patent protection fills a gap outlined above in software IPRs, allowing for more efficient protection of software than copyrights or trade secrets alone. Now, a company can publicly disclose certain parts of its software, previously only protected through software obfuscation and copyright protection. This disclosure is the aegis behind the patent system in general; by contributing innovation to the public domain, the patent holder gains a limited time to exclude other from using his invention.

c. The Open Source Conundrum—a Cautionary Note

In response to software patents, its critics have created a type of license called the open source software (OSS) license. Although OSS is often seen as a panacea of free software, saving a company some of the costs associated with software development, this software is not free in the classic sense. Rather OSS licenses present a myriad of problems to the software patent holder.

This is because the use of OSS requires acceptance of licensing provisions.[xxiv] These licenses can range from requiring software integrated with the OSS to be licensed under the same OSS license therefore forcing the integrated software into the public domain, to requiring compulsory patent licenses.[xxv] As a result, use of OSS can have a negative impact on a business' IP portfolio, especially if a company does not know OSS was used or did not examine the OSS license.[xxvi] These problems have only been exacerbated with the release of the GNU Public License version 3.

In general OSS licenses can be divided into four main types: those with few restrictions, infectious licenses that require all derivative works to have the same license, those that require patent licenses for use of the code, and those that contain patent retaliation clauses. The first type of license is one that contains few restrictions and usually only requires that the OSS author be given credit for his contribution. This type of license is considered benign and usually has no impact on the commercial viability of the code as a software product. Conversely, the other three types of licenses can have a potential negative commercial impact on a company's IP portfolio.

An infectious license is one that requires that all software code integrated with the OSS code be licensed under the same open source license.[xxvii] The infectious license also usually requires that the integrated code be made publicly available, essentially putting it in the public domain.[xxviii] However, for this type of license to apply, the OSS must be integrated directly into the developed software.[xxix] In general, statically linking code is considered to be integrating the code where as dynamic linking is not considered integration.[xxx] Commonly known infectious software licenses include the General Public License (GPL), the Artistic Licenses (Perl Licenses), and the Vim Licenses.[xxxi]

Another type of open source license is one that contains a compulsory or automatic patent licensing agreement.[xxxii] This license stipulates that use of the OSS code requires the user grant the creator of the OSS a patent license.[xxxiii] This patent license covers any patent which the user may own and which may be infringed by the OSS; it essentially requires any user of the code to agree not to sue the OSS creator for patent infringement for any patent upon which the code may infringe. In real terms, this means that a given patent portfolio upon which the OSS code could infringe would be granted an automatic patent license just by integrating the OSS code. Common licenses of this type are the Apache 2.0 Licenses.[xxxiv]

Finally, the last type of OSS license contains a patent retaliation clause and is considered the broadest type of OSS license.[xxxv] This clause states that a user may not sue any contributor to the OSS for infringement of any software patent without losing the OSS patent license. Regardless of the patent being infringed or the level of contribution, no software patent may be enforced by any OSS user against any contributor who infringes. Common licenses of this type are the Common Public License, the Firebird Public License, and the third version of the General Public License.[xxxvi] However, the GPL v.3 goes far beyond simply implicating the patent portfolios of those who contribute to OSS.

The GPL v. 3 is so software-patent-adverse that it implicates those companies who only "convey" software covered by this license. That is, any company who distributes software, even verbatim copies of that software, also grants a patent license for that software. This means that any distributor of software licensed under GPL v.3 can not sue for its patent portfolio without out violating the licensing agreement. This is particularly worrisome as GPL v.3 and previous versions of the GPL licenses are incompatible.

Unknowing or uninformed use of these licenses can cause serious problems for a business's IP portfolio. First, since a license is an agreement not to sue for a copyright violation, ignorance of the license is not an excuse.[xxxvii] Second, open source is an attractive alternative to full scale development as it often saves time and money.[xxxviii] Small companies are often tempted to use this software to have shorter development times and save on development costs. Engineers at large companies who are unaware of the effects of using open source may incorporate it into software products to reduce the development cycle.

To mitigate the risk of open source, engineers and software programmers need to understand the risks associated with open source use. For example, it may be possible to avoid the open source license through dynamically linking to the open source software instead of statically integrating the OSS into the software product. By dynamically linking, the code would not be fully integrated and a derivative work based on the OSS would not be created. In this way, dynamic linking could avoid not only infectious code but also certain patent licensing issues. Conversely the business may not have any patents or associated IP that would cover the OSS software and would not risk licensing any of its software patents.

3. Other Intellectual Property Protection

a. Integration of Intellectual Property Species in General

Literature and presentations on IP strategies, IP valuation, and other IP topics almost always address patents and patent portfolios. This focus on patents, however, overlooks the fact that legal protection of innovation of any kind, especially in high-tech fields, requires the use of more than one IP category. This overlap assures dual or multiple protection.

Professor Jay Dratler, in his *Intellectual Property Law: Commercial, Creative, and Industrial Property*, was the first to "tie all the fields of IP together."[xl] According to Dratler, IP rights, formerly fragmented by specialties, are now a "seamless web" due to progress in technology and commerce. Subsequently, the authors of Intellectual Property in the New Technology Age also stressed the need to "avoid the fragmented coverage . . . by approaching IP as a unified whole" and by concentrating on the "interaction between different types of IP rights." Thus, we now have a unified theory of IP management, a single field of law with sub-sets, and a significant overlap for the same IP or for different aspects of the same IP. Not taking advantage of the overlap misses opportunities and, according to Dratler, amounts to "malpractice."

Especially for high-tech products, trademarks and copyrights can supplement patents, trade secrets, and mask works ("blueprints" used in the R&D and production of semiconductor chips). One IP category, often patents, may be the center of gravity and more important than others. Other IP categories are then supplemental but equally valuable, as they may function to

- cover additional subject matter
- strengthen exclusivity
- invoke additional remedies in litigation
- provide a backup if a primary IP right becomes invalid

and thus provide synergy and optimize legal protection.

The most important, albeit most disputed, intellectual property management policy and strategy, is exploitation of the overlap between patents and trade secrets. There is of course no argument whatsoever about coexistence and compatibility of patents and trademarks. There is likewise no controversy whatsoever about franchise agreements which cover trademarks and trade secrets and constitute a huge category of hybrid license agreements. Nor is there polarity between copyrights and trade secrets; they can conjointly protect software.

b. How Software Patent Protection Complements Copyright Protection.

Following the acceptance of software as patentable subject matter in the U.S., companies extended software protection in ways that copyright protection did not allow. Instead of relying on exact replicas of proprietary code, derivative works, and trade secrets, software patents allowed protection when the same algorithm or method was used.[xli] Companies no longer needed to keep innovative software algorithms secret and fear another company would reverse engineer their code; as copyright law does not protect against reverse engineering the algorithm behind the functionality.

Further, as the intellectual property rights conferred by software patents and copyrights are not mutually exclusive, but rather dovetail, software can now be protected much more effectively. Patents protect the utilitarian contribution made to the art while copyright protects the software code from copying. That is, copyright protects the particular expression of the work, e.g. the source and object code, and patents protect one or more of the ideas behind the expression.

However, while copyright protects the entire software program from copying of the expression, the software patents protect only the ideas in the software covered by the patent. Ideas unprotected and disclosed or ideas already in the public domain may be freely copied as long as they are not also copies of a given expression of that idea. Yet, in the U.S., these unprotected ideas, including knowledge and know-how protectable via trade secrets, need not be disclosed to the public in order to gain copyright protection. Rather, the source code, in toto, may be copyrighted by submitting a portion of the code. Copyright Circular 61 gives the options for submitting software code with trade secrets; namely:

> Where a computer program contains trade secret material, [the registration must] include a cover letter stating that the claim contains trade secrets, along with the page containing the copyright notice, if any, plus one of the following:
>
> - Entirely new computer programs · First 25 and last 25 pages of source code with portions containing trade secrets blocked out; or
> - First 10 and last 10 pages of source code alone, with no blocked out portions; or
> - First 25 and last 25 pages of object code plus any 10 or more consecutive pages of source code, with no blocked-out portions; or
> - For programs 50 pages or less in length, entire source code with trade secret portions blocked out.
> - Revised computer programs · If the revisions are present in the first 25 and last 25 pages, any one of the 4 options above, as appropriate; or
> - If the revisions are not present in the first 25 and the last 25 pages:
>
> a. 20 pages of source code containing the revisions with no blocked out portions; or
> b. any 50 pages of source code containing the revisions with some portions blocked out.[xlii]

Following the USPTO allowance of software patents, many patents were aggressively pursued by companies to protect their software portfolios and more recently also business methods.[xliii] These patents were and continue to be used offensively in litigation to protect a company's innovations against others who independently create or appropriate the same invention.[xliv] The ability to protect software with patents allows for easier collaboration between software partners. Previously, joint research had to rely on non-disclosure agreements and derivative works for software protection. With a software patent, companies can simply license their innovations, which often makes the companies more likely to disclose their innovations.

c. How Software Patent Protection is Compatible with Trade Secret Protection

Trade secrets are the crown jewels of corporations. According to a 2003 survey on strategic IP management sponsored by the Intellectual Property Owners Association (IPO), patents are rarely viewed as an IP panacea, but rather as a supplement to other forms of IP protection.[xlv] Patents have limits, such as early publication, invent-around feasibility, and strict patentability requirements. Survey respondents did rate proprietary technology highly as a key source of competitive advantage, and a large majority of respondents (88%) cited skills and knowledge as the most important intellectual assets. Trade secrets are therefore directly implicated in the protection of proprietary skills and knowledge.

Moreover, patents are only the tips of icebergs in an ocean of trade secrets. Over 90% of all new technology is covered by trade secrets and over 80% of all license and technology transfer agreements cover proprietary know-how (trade secrets) or are hybrid agreements covering both patents and trade secrets. Very importantly, trade secret protection operates without delay and without undue cost against the world, while patents are territorial and so expensive to obtain and maintain that they can be acquired only in selected countries.

All patents are born as trade secrets: they precede patents, accompany patents, and follow patents. Patents and trade secrets are not mutually exclusive but actually highly complementary and mutually reinforcing; in fact they dovetail. The question is not whether to patent or padlock but rather what to patent and what to keep a trade secret and whether it is best to patent as well as to padlock, that is, integrate the patents and trade secrets for optimal synergistic protection of any innovation. Any contention that trade secrets cannot coexist with patents on a given invention because of the best mode requirement, overlooks the simple truths that this requirement applies:

- Only at the time of filing,
- Only to the knowledge of the inventors, and
- Only to the claimed invention.

In fact, the best mode requirement is actually no impediment to the coexistence of patents and trade secrets for almost any invention. Patent applications are normally filed very early, providing only rudimentary R&D data. Tom Arnold asserted that it is "flat wrong" to assume, as "many courts and even patent lawyer seem prone" to do, that "because the patent statute requires a best mode disclosure, patents necessarily disclose or preempt all the trade secrets that are useful in the practice of the invention." (1988 Licensing Law Handbook).[xlvi]

For example, almost every software program can consist of a combination of patentable material, copyrightable material and proprietary information and know how, i.e. trade secrets. At the simplest level, the copyright protects against literal copying of the code, in whole or part. As trade secrets are allowed to be redacted out of the copyrighted code, important innovations that are not disclosed in a patent application, for one reason or another, may be protected by trade secrets. This is also true of any collateral know-how.

The patented inventions, of course, fall under the protection of the patent act. Further, since patent applications need only disclose the best mode and enablement of the software innovation, the code itself and collateral know-how need not be disclosed, but can be maintained as trade secrets. The glue holding the patentable and trade secret ideas together, which further transforms these ideas into expression, is often the technical know-how, the grist for trade secrets. Software libraries, compilers, implementation schemes, and code optimizations techniques constitute just the tip of the iceberg in the average software company's know-how.

4. What about Sui Generis Protection?

Given the special nature of software, what is the best form of protection for software has been and still is a most unsettled and vexing, and hence very topical, issue in IP law and practice.

Congress, of course, did amend our copyright law in 1980 to make it clear that software is copyrightable. Likewise, legislation was enacted in foreign countries in past years, stipulating that software was only copyrightable, i.e. not patentable. TRIPS also required that copyright protection be provided by WTO countries. Thus, it was not surprising that Ralph Oman, the former Register of Copyrights and others maintained that an international consensus in favor of copyright protection had emerged,[xlvii] even though many believed that copyright protection was an "artificial construct," inasmuch as the aims of copyright law and computer programming are diametrically opposed, the former stressing subjective, individualistic, creative elements and the latter, objective, technical and scientific systematization. Software is functional, non-literal by nature as it performs a task or generates output.

Thus, there were many authors and practitioners here and abroad who believed that copyright law was inappropriate as forms of protection and it was patent law and/or *sui generis* systems which would offer better protection for software. Headlines of articles bore this out; to wit "The Case for Software Patent Protection,"[xlviii] "Software Patents Come of Age,"[xlix] "Patents, Not Copyright, Poised for Bigger Byte of Software."[l]

But D.A. Einhorn proclaimed in 1990:

> It is crystal clear that software is entitled to both copyright and patent protection. It should also be crystal clear that these forms of protection should not be mutually exclusive. There is no justification whatsoever in the Constitution, the federal statutes, or the case law to justify a denial of joint patent and copyright protection for software.[li]

In spite of that contention, many parts of the world are still debating whether or not software protection should or should not be extended from copyright to include patent as well.

However, there are significant problems with patenting software, in particular the statutory subject matter (algorithm) issue and the non-obviousness requirement, as illustrated by the following illustrious title: "Now You See it, Now You Don't: Was It a Patentable Machine or an Unpatentable 'Algorithm'?".[lii] And anent copyright, its shortcoming is well known as shown above and a copyright term for 100 years is incongruous for protection of ephemeral software programs.

With patent and copyright forms of protection being "Procrustean Beds," it should not come as a surprise that the notion of a *sui generis* form of protection for software in lieu of or in addition to present routes of protection had and still has considerable appeal.

Professor Samuelson's 1994 "*Manifesto Concerning the Legal Protection of Computer Programs*" comes immediately to mind,[liii] as well as Richard Stern's *sui generis* utility model law proposal, propounded in 1993[liv]. Indeed, I remember well that the first impulse by the IP profession back in 1965, when the issue first arose, was to provide a *sui generis* form of protection and *sui generis* IP protection is not uncommon at all. It exists for the protection of plants and artistic products. Moreover and very significantly, Congress, in 1984, fashioned the Semiconductor Chip Protection Act (17 U.S.C. § 901) to protect mask works for ten years in a

copyright-like manner. This was done hastily and improvidently according to some practitioners and this statute is apparently rarely used.

While the Intellectual Property Owners Association (IPO) and others are strongly in favor of a "unitary patent system," it can be argued equally or more forcefully that the slogan "one system does not fit all," is particularly well suited to protection of software, given its special, i.e. *sui generis* nature. Also, Congress had no hesitation to pass the Vessel Hull Design Protection Act of 1998 as *sui generis* protection (17 USC 1301 et seq.) and recently even passed broadening amendments. And protection for databases via a *sui generis* protection regime has also been under consideration.

Conclusion

Software patents have been shown to be an effective way to protect innovation in software. Software patents have proven to be enforceable in litigation and have been shown empirically to not limit innovation or entry into the software market. While OSS licenses present problems, software patent owners and allowable subject matter is continually changing; software patents remain a viable and effective way to protect innovations embodied in software. In conjunction with trade secrets and copyright protection, software patents have helped fill a void in the IPR system with respect to software. However, from a puristic point of view, properly drafted *sui generis* protection for software could provide more appropriate and less controversial protection for software.

Joseph J. D'Angelo
3rd year law student
Franklin Pierce Law Center
Concord, NH, USA

Karl F. Jorda
David Rines Professor of Intellectual Property Law
Director, Germeshausen Center for the Law of Innovation and Entrepreneurship
Franklin Pierce Law Center
Concord, NH, USA

---

[i] See Gottschalk v. Benson 409 U.S. 63, 175 U.S.P.Q. 673.
[ii] Id.
[iii] Diamond v. Diehr, 450 U.S. 175 (U.S. 1981).
[iv] Id.
[v] In re Beauregard, 53 F.3d 1583 (Fed. Cir. 1995).
[vi] 17 J. Marshall J. Computer & Info. L. 183, 195.
[vii] Id.
[viii] Id.
[ix] Id.
[x] 149 F.3d 1368 (Fed. Cir. 1998)
[xi] 1998 U.S. App. LEXIS 16869, *15.
[xii] Manual of Patent Examining Procedure § 2106.
[xiii] Recent Changes of Korean Intellectual Property Law, Bae, Kim & Lee (September 28, 2007).
[xvii] Eolas. Techs., Inc. v. Microsoft Corp., No. 99 C 0626 (N.D. Ill. Dec. 29, 2000).
[xviii] Renee BOUCHER FERGUSON, EPLUS WINS $37 MILLION PATENT INFRINGEMENT SUIT AGAINST ARIBA (2006), http://www.eweek.com/article2/0,1759,1764676,00.asp (last visited Dec. 10, 2006).

[xix] STACY COWLEY, HP TO PAY $325 MILLION TO SETTLE LAWSUITS WITH EMC, CIO (2005) http://www.cio-asia.com/ShowPage.aspx?pagetype=2&articleid=1041&pubid=5&issueid=36 (last visited Dec. 10, 2006).
[xx] Id.
[xxi] Id.
[xxii] Id.
[xxiii] RICHARD STALLMAN, SOFTWARE PATENTS – OBSTACLES TO SOFTWARE DEVELOPMENT (2002), http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html (last visited Dec. 10, 2006); IS THE US PATENT SYSTEM ENDANGERING AMERICAN INNOVATION? (2005), http://www.athenaalliance.org/aevents/patent_reform.html (last visited Dec. 10, 2006)(referencing David Kappos's speech on the software patents and needed change in the industry).
[xxiv] Open Source Initiative, The Open Source Definition, http://www.opensource.org/docs/definition.php (accessed Feb. 6, 2007).
[xxv] Adam Kubelka & Matthew Fawcett, No Free Beer – Practice Tips for Open Source Licensing, 22 Santa Clara Computer & High Tech. L.J. 797, 810-14 (2006).
[xxvi] Id.
[xxvii] See Andrew LaFontaine, Student Author, Adventures in Software Licensing: SCO v. IBM and the Future of the Open Source Model, 4 J. Telecomm. & High Tech. L. 449 (2006) (discussing the effects of different types of OSS licenses).
[xxviii] Id. at 462-63.
[xxix] Id. at 456 (discussing what integration is with respect to accepting a license).
[xxx] Id.
[xxxi] Id; Free Software Foundation, GNU General Public License, http://www.gnu.org/copyleft/copying-1.0.html (accessed Feb. 6, 2007).
[xxxii] See Open Source Initiative, Mozilla Public License 1.1, "Section 2," http://www.opensource.org/licenses/mozilla1.1.php (accessed Feb. 6, 2007) (providing an example of an OSS license granting a reciprocal patent license).
[xxxiii] LaFontaine, 4 J. Telecomm. & High Tech. L. at 462-63.
[xxxiv] See Open Source Initiative, Apache Software License, http://www.opensource.org/licenses/apachepl.php (accessed Feb. 6, 2007).
[xxxv] See Free Software Foundation, GPLv3, 2nd discussion draft, http://gplv3.fsf.org/gpl-draft-2006-07-27.html (accessed Feb. 6, 2007). (containing a patent retaliation clause).
[xxxvi] See Steven J. Vaughan-Nichols, Lawyers Express GPL 3 Concerns, http://www.eweek.com/article2/0,1759,1912999,00.asp (accessed Feb. 6, 2007) (explaining how GPL v.3 limits IP rights); Free Software Foundation Europe, Transcript of Richard Stallman at the 5th international GPLv3 conference; 21st November 2006 http://fsfeurope.org/projects/gplv3/tokyo-rms-transcript (accessed Feb. 2, 2007).
[xxxvii] Melville B. Nimmer & David Nimmer, Nimmer on Copyright § 10.01 n. 73.1 (Matthew Bender 2006).
[xxxviii] Kubelkal, 22 Santa Clara Computer & High Tech. L.J. at 799-803.
[xl] Jay Dratler, "Intellectual Property Law: Commercial, Creative, and Industrial Property," vii, (Law Journal Seminars-Press v.1 1991).
[xli] See 4 J. on Telecomm. & High Tech. L. 449 (commenting on copyright protection for software code).
[xlii] Circular 61, http://www.copyright.gov/circs/circ61.html (last visited November 7, 2007) (showing the different ways of registering software code and acknowledging that trade secret portions may be blocked out when registering the code).
[xliii] JAMES BENSON ET AL., AN EMPIRICAL LOOK AT SOFTWARE PATENTS (2004). http://www.researchoninnovation.org/swpat.pdf (last visited Dec. 10, 2006)(stating that software patents now make up 15% of patent filings).
[xliv] PAUL MCDOUGALL, HOW TO AVOID THE PATENT TRAP, INFORMATION WEEK (2006) http://www.informationweek.com/management/showArticle.jhtml;jsessionid=JPLHX3IU3BK2UQSNDLQCKH0CJUNN2JVN?articleID=193402991 (last visited Dec. 10, 2006).
[xlv] 2003 Survey on Strategic IP Management by the Intellectual Property Owners.
[xlvi] 1988 Licensing Law Handbook, Arnold, White, & Durkee, 37,. Appendix C, p. 295, 308 (Clark Boardman).
[xlvii] Oman, Ralph (1995), "Copyright and Software: The Emerging International Consensus", IPL Newsletter V.13 N. 3
[xlviii] "The Case for Software Patent Protection," 14 Hastings Comm/Ent L.J. 315 (1992).

[xlix] Yoches, E Robert and Hines, Doris Johnson, “Growing Pains: Software Patents Come Of Age,” The American Lawyer’s Corporate Counsel Magazine (1996).
[l] Tryzna, Peter K., “Patents, Not Copyright, Poised for Bigger Byte of Software” The Intellectual Property, Vol 1, No. 10 (July 1995).
[li] Einhorn, D.A. (1990), “Copyright and Patent Protection for Computer Software: Are They Mutually Exclusive?”. The Journal of Law and Technology, 278.
[lii] Peterson, R. M., “Now You See It, Now You Don't: Was It a Patentable Machine or an Unpatentable "Algorithm"?, On Principle and Expediency in Current Patent Law Doctrines Relating to Computer-Implemented Inventions” (1995) 64 Geo. Wash. L. Rev. 90.
[liii] Pamela Samuelson et al., “A Manifesto Concerning the Legal Protection of Computer Programs,” Colum. L. Rev. 2308 (1994).
[liv] Sterne, Richard H., “A Sui Generis Utility Model Law as an Alternative Legal Model for Protecting Software,” 1 U. Balt. Intell. Prop. L.J. 108 (1993).