

Open Source Licensing and Patent Strategy

Howard C. Anawalt¹

“Open Source Licensing” is one viable way of handling the complex intellectual property issues related to software. The following discussion examines the general legal and practical framework of Open Source. Particular attention is paid to patents and patent planning to aid the success of an Open Source strategy.

The basic approach of Open Source can be illustrated by a hypothetical company. Assume that Multi-Go has developed a powerful language translation program called “Flexi-Go.” Multi-Go decides that it will be best able to survive if many users use its product. Flexi-Go is a program which will need successful adaptation to various computer platforms (Mac, Windows, Linux, etc.) and to different word processing programs (Word, Word Perfect, Nisus, etc.) In addition, it is advantageous if Flexi-Go can adapt to subsets of language that need translation--legal usage, business terms, literary references. Multi-Go’s management may say: “If only many developers will choose Flexi-Go and adapt and enhance it, then it will be always flexible and will win new customers!” One of the business managers answers: “Let us freely license to all those who will agree to share their new developments with everyone else. That way, people will always look to Multi-Go as the leader, because they can always get the latest developments.” In short, the business model proposed is to allow an open license to all who will in exchange offer an open license to all future users and developers.²

1 Howard C. Anawalt is Professor Emeritus at Santa Clara University. He is author of H. C. Anawalt, *IP Strategy: Complete Intellectual Property, Planning, Access, and Protection* (Thomson/West). He was Invited Researcher at the Institute of Intellectual Property of Japan (IIP) in 2000-2001. He writes and consults on intellectual property in Silicon Valley, California, and Seattle, Washington. The author acknowledges the comments and suggestions of Mr. Joel Riff of GCA Law Partners LLP, Mtn. View, California, Hugh Treanor of Palo Alto, California, David A. Wheeler, software commentator.

2 The Multi-Go approach operates on a principle of reciprocity: the licensor releases its software source code and rights in exchange for a similar release by the licensee. Not all open source licenses require reciprocity. For example, the Berkeley Unix system is distributed under a very simple license that does not require reciprocity. See <http://www.opensource.org/licenses/bsd-license.php>. The popular Apache program is similarly distributed. <http://www.opensource.org/licenses/apachepl.php>. One software commentator observed in a letter to the author: “many OSS projects have a different business model. e.g., many operate as consortiums of companies, where the idea is not to gain money per se, but to share costs and thus reduce the outflow of

Open Source licensors and licensees choose an approach that is calculated to support business by relatively non-restrictive licensing practices in an environment that bristles with thorny and difficult intellectual property issues. As a strategy, its success hinges on minimizing the risks that hostile intellectual property claims will be made against itself or its licensees.

Why Is It Called “Open Source?”

Software producers enjoy two types of protection--legal doctrines and practical steps. A producer may hinder both copying and simulation of its software by the simple expedient of *concealing the source code*. This is in effect the first line of defense. The second line of defense is the use of the wide range of intellectual property doctrines that apply to software.

Developers often wish to write new software that is compatible with or competes with existing software.³ Compatibility and competition are clearly legitimate goals. To achieve either, the developer will usually need to study the existing software. In order to study the software easily and effectively the developer needs access to the human readable version, the *source code*. Many software producers make it very difficult to study their programs because they release only the *object code* of their programs. Object code is the version of the software that operates on the machine. The human readable version or source code is kept secret. One can derive the way the source code works by studying the object code (reverse engineering) or from conducting a clean room operation.⁴ However, these methods are laborious and thus

money. Money is still involved, at least in some sense, since a penny saved is a penny earned. But x.org, Apache, Mozilla, etc. don't really make money as much as provide a forum so contributors can save money/time.”

3 For example, one developer may wish to achieve compatibility by writing a program that runs on a particular operating system or machine. Another may wish to compete by producing a computer game that is more attractive than an existing game. Still others will wish to write enhancements for existing programs.

4 In the United States, the Committee for Interoperable Systems was formed in the 1980s to promote access to software. One focus was to allow for legal reverse engineering. Two attorneys, Mr. Peter Choi and Mr. Jonathan Band, and others paved the way to such access by successfully establishing a general privilege for limited copying for the purpose of “reverse engineering” of software. See *Sega v. Accolade*, 977 F.2d 1510 (9th Cir. 1992) and *Lexmark Intern., Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 549 (C.A.6 (Ky.), 2004). H. C. Anawalt, IP Strategy § 1:71 and 1:93. Reverse engineering can be frustrated by “click wrap” licenses that create contractual limits. See the *Lexmark* case and *Bowers v. Baystate Technologies, Inc.*, 320 F.3d 1319 (Fed. Cir. 2003).

detract from the primary work of designing the new software.⁵

The Open Systems movement seeks to eliminate both the practical and legal difficulties of studying and using the software. Source code is made available to others and legal restrictions on use are greatly reduced through licenses. The developers who opt for this approach appear willing to compete rather than exercise control by copyright and patent. They are willing to rely primarily on such things as product flexibility, cooperation, and reliability. They need not forsake all intellectual property invention protection, but rely on it much less. For instance an open source developers may rely on trade secret protection especially during development of new products or features. In addition they may rely on name recognition, which is, of course, aided by trademark law. Depending on the specific approach chosen, the open source developer may also rely on both copyright and patent protection. However, to the extent that the developer enforces strong control through these legal doctrines, it departs from both the spirit and the advantages of the open source approach.

Legal Protections of Software

Software is protectable by a comprehensive array of intellectual property legal doctrines.⁶ First, in the United States and many other countries, software is immediately protected by copyright as soon as it is written down or “fixed” in any reasonably permanent medium, for example, computer memory. The work does not have to be new or inventive. It need only be “original” in the sense that the expression comes from the author, rather than from some obvious arrangement such as the alphabetical organization of a telephone book.⁷ Protection is limited to the “expression” or particular articulation in the software. The protection does not extend to the underlying ideas or system of the software.⁸ Under United States law in most

5 “Reverse engineering” involves decompiling software then studying how it works. With a “clean room,” the software is studied by one set of engineers to determine its necessary specifications, then a separate set of engineers designs code that will produce the desired software performance. See H. C. Anawalt, *IP Strategy* § 4:22. Reverse engineering may also subject to the claim it involves a degree of interim copying that violates terms of the licensing contract that gives the developer access to the software. See *Bowers v. Baystate Technologies, Inc.*, 320 F. 3d 1319 (Fed. Cir. 2003).

6 The discussion is limited to United States law. To the extent possible, Japanese colleagues have added relevant references to Japanese law, practices, and customs.

7 *Feist Publications, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 350-53, 111 S.Ct. 1282, 113 L.Ed.2d 358 (1991).

8 See 17 U.S.C. 102 (b), *Bateman v. Mnemonics, Inc.*, 79 F. 3d 1335 (5th Cir.) and *Lexmark Intern., Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 546 (C.A.6 (Ky.), 2004.)

instances one must register the software with the Register of Copyrights before the software can receive legal protection in the Courts. The copyright exists, it simply cannot be enforced until registered.

Software is also protected by the “Digital Millennium Copyright Act” (DMCA). The DMCA provides that so long as some portion of a work contains some copyrighted material, it can be protected by a “digital lock.” That is, any safeguard such as the common device of a password can be enforced by federal civil and criminal remedies.⁹

Trade secret law also protects software to the extent that particular matters are kept secret. In most United States jurisdictions, one who claims he has a secret must keep it from falling into the hands of others. In essence, this branch of law protects confidentiality.¹⁰ The open source developer will no doubt wish to guard many matters during the process of development. In fact, companies that agree to cooperate by sharing software through open source licensing may place a very high priority on keeping their developments secret until they are ready to release them, so they can gain marketing and name recognition values. Later, when it comes time to release the product, the open source developer may choose to abandon much trade secret protection.

Finally, software can be protected by patents, so long as it meets the traditional requirements. The claimed invention in the software must be new, useful, and non-obvious, and it must govern some process that the software implements.¹¹ While software can be freely patented in the United States, it does have limitations. One of

9 “Digital Millennium Copyright Act” (DMCA), 17 U.S.C. 1201 and following.

10 Trade secret law is generally governed by the rules of each of the fifty states in the United States. To a certain extent, trade secrets have been made a subject of federal protection in the United States by the “Economic Espionage Act of 1996,” 18 U.S.C. §§1832-1839. The DMCA also effectively protects trade secrets. So long as a digital batch of data contains *some* increment of copyrighted material, the whole may be protected by a digital lock under the DMCA. See H. C. Anawalt, IP Strategy §§ 1:56, 1:148, and 1:158.

11 The claims must implement a process, rather than state a general principle or “mathematical algorithm.” See *Gottschalk v. Benson*, 409 U.S. 584 (1979), *Arrhythmia Research Technology v. Corazonix Corp.*, 958 F. 2d 1053 (Fed. Cir. 1992), *State Street Bank and Trust v. Signature Financial Group, Inc.*, 149 F. 3d 1368 (Fed. Cir. 1998). “The Supreme Court has interpreted (the) statutory range of patentable subject matter to be quite broad, but hardly universal....That wide scope nevertheless excludes laws of nature, natural phenomena, and abstract ideas.” *Smithkline Beecham Corporation v. Apotex Corp*, 403 F.3d 1331, 1361 (Fed. Cir. 2005), concurring opinion. See H. C. Anawalt, IP Strategy § 1:25.

the primary ones is that the inventive aspect can not be “obvious.” That is, the claimed invention must not present a solution that would be fairly obvious to other people who work with software.¹² Due to the nature of software, much that is developed would likely be reasonably obvious to a person “having ordinary skill in the art.”

The Critical Role of Patents

Usually developers who choose the Open Source strategy agree to a contract which includes a comprehensive license that grants copyright permissions. The license will grant access to the source code and may provide that new developments by the licensee are similarly open to use and development by others. Under one widely used approach, the parties provide that adaptations and enhancements must be similarly open to future developers.¹³ The license may include necessary or useful provisions concerning reverse engineering, non-disclosure, protection of trade secrets, warranties, indemnification, and such matters as dispute resolution and selection of courts and governing law. The license may include permissions to use passwords and so forth to avoid difficulties under the DMCA.

Let us assume that the entire licensing arrangement satisfies the parties. They have succeeded in opening the software. If they have not considered the force of patents, however, the analysis will be incomplete and the plan will be vulnerable.

A patent is the most potent form of intellectual property. Once granted, the scope of the patent grants strong intellectual property claims that resemble the description of ownership of a piece of real property. Copyright leaves matters open for later interpretation: What is the protected expression? Is the copying permitted by fair use? Patent avoids much of this uncertainty. By contrast, one can read a patent in advance and understand more or less exactly what it covers. When a patent is tested in court, more limited judicial review occurs than in a copyright case. A patent’s scope and validity can be tested in any District Court. The appellate jurisdiction of the patent questions lies with the Federal Circuit. That court declares the patent law, subject to review by the Supreme Court. In general, claims construction and “obviousness” are questions of law.¹⁴

12 35 U.S.C. 103. *Graham v. John Deere, Co. of Kansas City*, 383 U.S. 1 (1965).

13 This reciprocal approach takes the position that any party should be free to redistribute the open source software, should have access to the source code, and should be bound to allow similar distribution of new derived works. “The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.” See the Open Source Initiative, <http://opensource.org/docs/definition.php>. The Open Source Initiative website offers many versions of open source licenses for review and adoption.

14 A patent’s scope and validity can be tested in any District Court.

An analogy may be drawn to the game of chess. All the doctrines of intellectual property offer advantages. However, because of its strength and clarity, a patent operates as the Queen on the chess board. When a relevant patent appears it dominates.

Thus, suppose that our original hypothetical company, Multi-Go, as entered into a comprehensive open source agreement and license with another developer, Grammar-Scout. If a third party comes forward with a patent that covers an important part of the developments of either party, the Multi-Go/Grammar-Scout arrangement can be stopped in its tracks!

The emphasis here and throughout the remaining discussion will be on software patents that cover some basic element, approach, or field of application of the open source software. Examples of such broad patents include file compression software, computerized analysis of electrocardiograph signals, and computerized accounting systems.¹⁵ In general, it is only the broader patents that will disrupt an Open Source development. A developer may be able to design around more confined patents.

No agreement between parties can limit the rights of non-parties.¹⁶ Thus, an open source license will not eliminate third party claims. Any party who claims

However, the appellate jurisdiction of the patent questions lies with the Federal Circuit. That court declares the patent law, subject to review by the Supreme Court. Thus, patent law achieves uniformity. In general, claims construction and “obviousness” are questions of law. *Bowers. v. Baystate Technologies, Inc.*, 320 F.3d 1317, 320 F.3d 1317, 1323 (Fed Cir. 2003) cert. denied 539 U. S. 928 (2003). In copyright cases, however, each of the federal circuits may vary in interpretations, making it more difficult to arrive at definitive rules.

15 Sperry Corporation patent number 4,558,302, "High Speed Data Compression and Decompression Apparatus and Method" (data compression), *Arrhythmia Research Technology v. Corazonix Corp.*, 958 F. 2d 1053 (Fed. Cir. 1992) (electrocardiograph), *State Street Bank and Trust v. Signature Financial Group, Inc.*, 149 F. 3d 1368 (Fed. Cir. 1998) (accounting system). “There are a number of data compression algorithms in use....UNIX systems come equipped with programs called compress and uncompress that use the Lempel-Ziv-Welch data compression algorithm to do their thing. Unfortunately, there are patent issues with this algorithm, and so the GNU people came up with their own compression algorithm that is embodied in the gzip program....”

Mark Hays

<http://math.arizona.edu/~restrepo/481/msg00054.html>

16 Benefits can be conveyed to third parties by contracts, but obligations cannot bind anyone not party to the contract. “It goes without saying that a contract cannot bind a nonparty.” *EEOC v. Waffle House, Inc.*, 534 U.S. 279, 122 S.Ct. 754, 764, 151 L.Ed.2d 755 (2002).

copyright, trade secret, patent, or another violation remains free to sue all parties. Among these claims, patent stands out, because its scope is spelled out in the patent itself, and a patent claim covers all uses of the matter described. Patent claims are not defeated by fair use as copyright claims are. Patent claims are not undermined by inadvertent disclosures as trade secrets are. Patents are by their nature open, disclosed to all, and preemptive of the technologies that they cover.

An open source developer has a range of choices on how deal with a hostile patent claim: “Shall we resist the claim? Shall we redesign the product? Shall we adopt a strategy that involves both?”¹⁷ In deciding how to deal with patent claims one needs to assess the strength of the claim and the legal resources required to resist the claim.¹⁸ For example, the defendant may have a good argument that a claimed invention is obvious. However, proving this point can involve huge expense. The law suit can drag on, customers may be lost, progress may be disrupted by depositions and court proceedings, vital employees may leave, etc. It is a sad fact that it is not necessarily the superior legal argument that may win a case. The case may turn on the ability of one side to spend more money on the litigation.

The open source developer does have some advantages when faced with patent claims. The claims themselves must be presented in the patent. The developer can study them and build a clear approach for redesign. If the patent claims appear to be weak or their scope can be narrowed, the project may be able to move along smoothly while negotiations occur. In addition, the fact that the software is open source may bring in other developers as allies in resisting or limiting the scope of patent claims.

Preventing Hostile Patent Claims

The fact that software can be patented actually offers a powerful tool for individual Open Source software developers and the Open Source movement. The

17 When faced with a patent claim, an alleged infringer faces a spectrum of choices, rather than dichotomy--redesign or go to court. One can resist the claim and engage in redesign at the same time. The scope of the patent claims may be narrow, allowing minimal redesign. It is a question of assessment and deciding on resources and expenses. Some companies with large software patent portfolios have announced policies of non enforcement against certain Open Source projects. An IBM press release on January 11, 2005 announced “IBM today pledged open access to key innovations covered by 500 IBM software patents to individuals and groups working on open source software....The pledge is applicable to any individual, community, or company working on or using software that meets the Open Source Initiative (OSI) definition of open source software now or in the future.”

18 An alleged infringer should make a careful initial assessment of the validity of the claim in order to avoid being found liable for willful infringement, in which case one may be liable for treble damages. 35 U. S. C. 284.

dominant form of software protection is copyright. However, as we have seen, a relevant patent changes the dynamics completely. The patent holder has the more forceful position and may be able to call the shots. However, if a software process is in fact subject to a patent it is inconsistent to say that it is also an expression that can be controlled by copyright. Patents provide ownership of a process. Copyright, however, does not permit ownership of processes or systems.¹⁹ If a function of software is patented, that function should not also be subject to copyright protection.²⁰

We can extend this analysis as follows: If a software design, routine, or organization of lines of codes is subject to being patentable, it ought also be viewed as functional to such a degree that it is no longer mere expression. Thus, if the Open Source developer can prevent a patent from issuing on patentable software, that developer accomplishes two things at once. It protects itself from patent claims. It also reduces the likelihood of a successful copyright claim.

Screening and Strategy Regarding Patents

Simply disclosing a technology provides one means of reducing the likelihood of an invention from being controlled by a patent.²¹ A disclosure may prevent the technology from being “novel” when someone else applies for a patent on that technology. Alternatively, the disclosure may render a related, but different technology “obvious.”

Simple publication may not suffice, however. For example, a printed publication does not prevent a patent from issuing unless the publication predates an

19 “In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.” 17 U.S.C. 102 (b). “This provision embodies the common-law idea-expression dichotomy that distinguishes the spheres of copyright and patent law. ‘[U]nlike a patent, a copyright gives no exclusive right to the art disclosed; protection is given only to the expression of the idea--not the idea itself.’...” *Lexmark Intern., Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 534 (C.A.6 (Ky.), 2004). See H. C. Anawalt, *IP Strategy* § 1:73.

20 The logic of the statutes and cases should preclude dual protection, however, the author knows of no court specific court holding on the point.

21 “The strategy of “defensive publication” is an old one in patent law, perhaps best exemplified by IBM’s longstanding practices in this area. IBM has long published a “Technical Disclosure Bulletin” aimed at precluding other firms from obtaining patents on technical advances that IBM itself chooses not to patent. [FN34] Other firms followed IBM, and a publication called “Research Disclosure” was even launched to facilitate the practice.’ Robert P. Merges, *A NEW DYNAMISM IN THE PUBLIC DOMAIN*, 71 U. Chi. L. Rev. 183, 194, Winter 2004.

application for a patent by one year.²² In such cases, the patent law provides an affirmative method of disclosure, filing a “statutory invention registration.”²³ The process is simple. One registers a patentable invention with the Patent Office and has it published. The registration requires that one describe the invention and its best mode of operation, waive a patent, and pay application and processing fees.²⁴ A published invention registration operates as a patent for all purposes, except enforcement. It is, in essence, a disclosure to the public that defends against patent claims, but does not allow claims of infringement. The registrant does not have to pay the maintenance fees required for a patent.

Developers have excellent options for using patent rules to enhance the success of an Open Source strategy.²⁵ A coordinated plan for using patent rules to support the strategy would include the following elements:

1. *Innovation screening and disclosure.* Developers should routinely screen software development for potential patentability of *major* software developments. The process need not be burdensome.²⁶ As noted earlier, the open source developer should be primarily concerned with patents of broad scope that may cut deeply into its software project. Thus, the individual developer should look primarily for patentable aspects that may have a broad impact on how his software works (e.g. file compression) or what field of business or enterprise the software will serve (e.g. accounting software.) Larger developers might adopt a somewhat broader patent screening guideline that goes beyond their particular product line.²⁷

22 35 USC 102 (b). William G. Phelps, “When does on-sale bar of 35 U.S.C.A.102(b)... prevent issuance of valid patent,” 155 A.L.R. Fed. 1.

23 The procedure is governed by 35 USC 157.

24 35 USC 157 and 35 USC 112. See also, 15 Moy on Patents 135.

25 As noted earlier, the Open Source approach or “movement,” is not one thing, but represents a range of approaches. Here are some useful websites for reference:

1) David A. Wheeler, comments on Open Source http://www.dwheeler.com/oss_fs_why.html (Czech | French | Japanese | Spanish translations available online) provides valuable comments on market share, reliability, etc. 2) MIT collection of papers, http://opensource.mit.edu/online_papers.php. 3) The Open Source Initiative, <http://opensource.org>. 4) A very wide open license style, “the BSD license” --<http://www.opensource.org/licenses/bsd-license.php>. 5) “Copyleft” and the GNU license, <http://www.gnu.org/copyleft/copyleft.html>.

26 Developers need to avoid burdensome aspects. Being too fussy can discourage the overall process. It ought to be kept as simple and straightforward as possible.

27 Software is often huge. It may contain thousands of lines of code.

The screening would be orienting toward making the following choices concerning patentable aspects:

- Preserve secrecy for trade secret purposes.²⁸
- Disclose only as part of a licensing program (e.g. disclose to open source licensees.)
- Disclose broadly through publication.
- File for a preemptive disclosure (“statutory invention registration”) under 35 U. S. C. 157.
- Apply for and obtain a patent and pay on-going maintenance fees.

One advantage of obtaining patent is that those who obtain improvement patents will need to obtain a license to practice their improvements.

2. *Cooperation and strategy regarding patents.* Some cooperation already exists among Open Source developers on legal questions. This is evidenced by standard licenses, websites, conferences, and various general practices. Developers can take further steps to use intellectual property law to favor the Open Source environment. Strategies can be designed to minimize, prevent, and deal with legal challenges to Open Sourcers.

Litigation is expensive, and patent litigation is more expensive than most others. Minimizing patent claims provides a good point of focus for cooperative efforts. Patent claims provide a more defined set of questions that do copyright or trade secret. Thus, agreement on courses of action will be more manageable. Successful legal defense against patent claims can have a broad effect on clarifying the important dividing line between functional works and those that present mere “expression.” If patentable works can be either excluded or disclosed, then open source approaches may be keep truly open by use of software licenses.

Cooperative action may include the following:

1) Decisions on patents. Deciding which inventions to disclose and which to patent. For instance, an innovator might choose to patent a ground breaking innovation precisely in order to be able to defend the Open Source approach.

2) Cooperation on defense against patent claims. Developers can coordinate on patent litigation and agree on joint defense efforts. Legal resources and defense funds could be shared. Licenses might include a provision for a contribution to a legal defense fund.²⁹

Programmers and smaller software developers would be overwhelmed if they pondered each segment of code.

28 Keeping important programming aspects secret runs contrary to the intentions of open source. Thus, it would be avoided by the true open sourcer, except during a limited development phase.

29 Some insurance companies may begin to offer insurance to cover claims against open source developers. InfoWorld has reported one company that plans to offer such insurance. http://www.infoworld.com/article/04/03/16/HNopeninsurance_1.html. Insurance, even if available, is a very different approach.

Conclusion

Open Source licensing offers a practical and sustainable approach to software development. That approach relies on release of developments to a broad public, rather than jealous protection of every aspect of intellectual property. Open Source development allows one to rely on trade secret, digital lock, patent, and trademark protections. However, those must be employed in a manner that is consistent with the open source design, if the strategy is to succeed. An Open Source developer does not obtain automatic protection against third party claims by entering into an open source agreement. In order to protect an Open Source project the developer needs to take into account the full range of intellectual property doctrines, including patent claims. With regard to patents, it is advisable for the developer to review projects to determine patentability of major aspects and to develop an overall patent strategy.