Licenses, contract protection for software [*]

I. Introduction

When software was first commercially marketed by a developer to a user, the protection used was a written agreement between the developer and the user that specified how the software could be used, such as on the user's own computers for the user's own purpose. It may have also indicated impermissible uses, like on the user's machines as a service to another user.

Contracts are essentially private law between the parties to the contract. The role of the government is not to set the terms of the agreement, such as the allowed uses for the software, but to enforce the terms of the contract if one of the parties *breaches*, or fails to abide by, the contract. The actual terms of the contract are set by the parties to it. Of course, the courts will not enforce a contract term that is contrary to law or goes against public policy, so there are effectively limits on the terms of a contract. For example, a court will not require payment of an illegal gambling debt, even though there is a contract agreeing to its payment. (However, the bookie may have other ways of "enforcing" the contract, perhaps by "taking out a contract" on the debtor.)

Traditional contract law required two distinct things for a contract to be formed. There had to be an agreement between the parties to the exact terms of the contract – a "meeting of the minds" – and there had to be something of value given or forgone by each party – "consideration." In the case of contracts greater that a particular value (often $500), whose performance takes an extended time (often a year or more), or involves a particular thing (like the sale of land), statutory law (called a "Statute of Frauds," from an old English law) may require that the contract be in writing and signed to aid in its enforcement in court. The trend has been to move away from the "complete meeting of the minds" requirement for the formation of a contract, where every term had to be agreed to by the parties, and go to "substantial agreement" for commercial transactions, where there are established rules for filling in the terms that were not specified in the contract or deciding which of two conflicting terms should prevail. This is the approach taken by the Uniform Commercial Code, statutory law that governs sales, leases, and other commercial transactions that has been adopted by all the states.[1]

Of course, since writings are inherently imprecise and possibly ambiguous, even if there has been a "meeting of the minds," it may be necessary at some later time for a court to interpret the meaning of a term in a contract. Over centuries, rules have come about for resolving such problems.

One common rule is that when contract language is reasonably open to two interpretations, the one less favorable to the person who supplied that language is preferred. That is because that is the person in the best position to provide clear and unambiguous language, and so should not benefit from not having done that. This is

[*] Copyright © 1999-2009 Lee A. Hollaar.

[1] Louisiana has not fully adopted the Uniform Commercial Code, because its legal structure is not based on the English common law but the civil code as in France. However, they have adopted most of its provisions.

particularly useful when trying apply a standard form contract to particular circumstances.

Another rule is that when there are two interpretations of a contract term and only one is in the public interest, that one is preferred.

Much as contracts that spell out the conditions for renting houses or apartments are called leases, contracts that grant the right to use intellectual property are called *licenses*. Licenses can be either *non-exclusive*, which means that the licensor (the owner of the intellectual property granting the right to use) can grant further licenses, or *exclusive*, which means that the licensor cannot grant any further licenses in the particular area of exclusivity (such as a particular geographical area, like a city or state, or a particular type of usage, such as in the mining industry). An exclusive license does not mean that that licensee is the only licensee in the area of exclusivity, only that there will be no further licensees in that area.

Licenses were a good fit for the first commercially-marketed software. There were questions whether other types of protection, such as copyright or patent were even possible for software. Since there were few computers outside of major companies or universities, it was reasonable to negotiate a contract between the developer and the user.

Licenses remain a mainstay in the protection of computer software. Licenses are important in protecting source code and other material not generally released to the public. Licenses also limit the warranty provided with computer software, specifying that certain warranties are limited or don't apply, and indicating what will be done if the software doesn't perform as specified. (Warranties are beyond the scope of this course.) Licenses also attempt to limit the user rights provided under copyright or impose special responsibilities on users.

II. Licenses that change the copyright rules

As we have seen, the copyright laws not only protect the copyrighted work, but allow a user to do certain acts, such as loading a program into RAM from the hard disk. As part of a license, can a user give up the rights he has under the federal Copyright Act?

The simple answer seems to be "yes, if the user wants to give up rights as part of a contract, why not?" However, there is a strong argument that Congress intended the Copyright Act to define the balance between copyright owners and users. In section 301 of the Copyright Act of 1976, Congress indicted that they were preempting all state laws that addressed the same subject matter as copyright.

> On and after January 1, 1978, all legal or equitable rights that are equivalent to any of the exclusive rights within the general scope of copyright as specified by section 106 in works of authorship that are fixed in a tangible medium of expression and come within the subject matter of copyright as specified by sections 102 and 103, whether created before or after that date and whether published or unpublished, are governed exclusively by this title. Thereafter, no person is entitled to any such right or equivalent right in any such work under the common law or statutes of any State.

This was primarily to eliminate the old dual system of copyright, where there was state protection until a work was published and federal protection after publication. In

House Report 94-1476, the Committee on the Judiciary explained the importance of this change.

> Section 301, one of the bedrock provisions of the bill, would accomplish a fundamental and significant change in the present law. Instead of a dual system of "common law copyright" for unpublished works and statutory copyright for published works, which has been the system in effect in the United States since the first copyright statute in 1790, the bill adopts a single system of Federal statutory copyright from creation. …
>
> By substituting a single Federal system for the present anachronistic, uncertain, impractical, and highly complicated dual system, the bill would greatly improve the operation of the copyright law and would be much more effective in carrying out the basic constitutional aims of uniformity and the promotion of writing and scholarship.

While the Committee was addressing the problems with the non-uniformity of the dual federal-state system of copyright protection, clearly many of the same problems of uncertainty about whether something is permitted or not are present where the rules are being defined by the license accompanying a particular work. It is hard to be diligent about respecting copyright when the rules are different for each work out of hundreds that you may be using.

The Committee also discussed the scope of this preemption.

> The intention of section 301 is to preempt and abolish any rights under the common law or statutes of a State that are equivalent to copyright and that extend to works coming within the scope of the Federal copyright law. The declaration of this principle in section 301 is intended to be stated in the clearest and most unequivocal language possible, so as to foreclose any conceivable misinterpretation of its unqualified intention that Congress shall act preemptively, and to avoid the development of any vague borderline areas between State and Federal protection.
>
> Under section 301(a) all "legal or equitable rights that are equivalent to any of the exclusive rights within the general scope of copyright as specified by section 106 are governed exclusively by the Federal copyright statute if the works involved are "works of authorship that are fixed in a tangible medium of expression and come within the subject matter of copyright as specified by sections 102 and 103." All corresponding State laws, whether common law or statutory, are preempted and abrogated. Regardless of when the work was created and whether it is published or unpublished, disseminated or undisseminated, in the public domain or copyrighted under the Federal statute, the States cannot offer it protection equivalent to copyright. …
>
> As long as a work fits within one of the general subject matter categories of sections 102 and 103, the bill prevents the States from protecting it even if it fails to achieve Federal statutory copyright because it is too minimal or lacking in originality to qualify, or because it has fallen into the public domain. …

> The preemption of rights under State law is complete with respect to any work coming within the scope of the bill, even though the scope of exclusive rights given the work under the bill is narrower than the scope of common law rights in the work might have been.

The Committee had previously indicated that section 301 does not preempt state protection of works that are not fixed in a tangible medium of expression, such as classroom lectures that are not recorded, because they are not the subject of the Copyright Act. Originally, they had planned to indicate examples of laws that would not be preempted, but that didn't make it to the final act.

> The examples, while not exhaustive, are intended to illustrate rights and remedies that are different in nature from the rights comprised in a copyright and that may continue to be protected under State common law or statute. The evolving common law rights of "privacy," "publicity," and trade secrets, and the general laws of defamation and fraud, would remain unaffected as long as the causes of action contain elements, such as an invasion of personal rights or a breach of trust or confidentiality, that are different in kind from copyright infringement. Nothing in the bill derogates from the rights of parties to contract with each other and to sue for breaches of contract; however, to the extent that the unfair competition concept known as "interference with contract relations" is merely the equivalent of copyright protection, it would be preempted.

But at the same time, the drafters of the Copyright Act of 1976 recognized that conditions regarding the transfer of a copy could be imposed by contract. In their discussion of the first sale provision of Section 109, they stated:

> Thus, for example, the outright sale of an authorized copy of a book frees it from any copyright control over its resale price or other conditions of its future disposition. A library that has acquired ownership of a copy is entitled to lend it under any conditions it chooses to impose. This does not mean that conditions on future disposition of copies or phonorecords, imposed by a contract between their buyer and seller, would be unenforceable between the parties as a breach of contract, but it does mean that they could not be enforced by an action for infringement of copyright. Under section 202 however, the owner of the physical copy or phonorecord cannot reproduce or perform the copyrighted work publicly without the copyright owner's consent.

And so we are back to the old question of whether the lawful possessor of a digital work can make the intermediate copies necessary to use the work without the permission of the copyright owner. If permission is necessary, then the copyright owner can set the terms of such permission in a license and enforce them as a copyright infringement. And that is what most "licenses" that accompany computer software and other digital works try to do, and in the process also try to limit user rights under the copyright law such as Section 109's first sale provisions and Section 117's archival, adaptation, and essential copying rights.

III. Two court cases, two different results

To the extent that the effect of section 301 on the scope of software licenses has been considered by the courts, it has been in the area of shrinkwrap licenses that commonly accompany mass-marketed software. Two appeals courts have considered the issue, and came to very different conclusions.

III.A. Vault v. Quaid

In *Vault Corp. v. Quaid Software,*[2] the Fifth Circuit affirmed a decision that the program developed by Quaid did not infringe Vault's copyright. Vault produced a program that prevented the copying of floppy disks protected by their program, while Quaid produced a program to copy the protected floppy disks. Because there was a substantial noninfringing use for Quaid's copy program – to produce archive copies permitted under section 117 – Quaid was not a contributory or vicarious infringer. However, Quaid had reverse engineered Vault's product to determine how the protection scheme worked, and that was not permitted under the shrinkwrap license that came with Vault's software.

While the legality of shrinkwrap licenses is questionable, the Louisiana Software License Enforcement Act had been passed in 1987, at the urging of software vendors. It was the first state law that recognized the validity of shrinkwrap licenses as long as certain requirements were met.

> Louisiana's License Act permits a software producer to impose a number of contractual terms upon software purchasers provided that terms are set forth in a license agreement which comports with [the Louisiana Act], and that this license agreement accompanies the producer's software. Enforceable terms include the prohibition of: (1) any copying of the program for any purpose; and (2) modifying and/or adapting the program in any way, including adaptation by reverse engineering, decompilation or disassembly. The terms "reverse engineering, decompiling or disassembling" are defined as "any process by which computer software is converted from one form to another form which is more readily understandable to human beings including without limitation any decoding or decrypting of any computer program which has been encoded or encrypted in any manner."

The Fifth Circuit then looked at whether this touches on an area covered by the federal copyright laws, and is therefore preempted. Their decision was simple.

> The provision in Louisiana's License Act, which permits a software producer to prohibit the adaptation of its licensed computer program by decompilation or disassembly, conflicts with the rights of computer program owners under §117 and clearly "touches upon an area" of federal copyright law. For this reason, and the reasons set forth by the district court, we hold that at least this provision of Louisiana's License Act is preempted by federal law, and thus that the restriction in Vault's license agreement against decompilation or disassembly is unenforceable.

[2] 7 USPQ2d 1281 (1988).

So licenses, or at least shrinkwrap licenses covered by a special state law and not general contract law, cannot preempt the user rights provided by the federal Copyright Act. That ended the push for special state legislation to legitimize shrinkwrap licenses. Only Louisiana and Illinois had adopted such laws, and both repealed them after this decision. For a while, it appeared to be accepted law that to the extent shrinkwrap licenses conflicted with federal copyright law, they were preempted. Their use was primarily to limit warranties.

III.B. ProCD v. Zeidenberg

The interest in shrinkwrap licenses revived after the Supreme Court decided *Feist v. Rural Telephone.*[3] In *Feist*, the Supreme Court held that a telephone white pages directory did not have sufficient originality to qualify for copyright protection. We will examine *Feist* in more detail when we look at database protection.

The question of whether federal copyright law preempts shrinkwrap licenses was considered by the Seventh Circuit in *ProCD Inc. v. Zeidenberg.*[4] As the Seventh Circuit explained the facts.

> ProCD, the plaintiff, has compiled information from more than 3,000 telephone directories into a computer database. We may assume that this database cannot be copyrighted, although it is more complex, contains more information (nine-digit zip codes and census industrial codes), is organized differently, and therefore is more original than the single alphabetical directory at issue in *Feist Publications, Inc. v. Rural Telephone Service Co.* ProCD sells a version of the database, called SelectPhone™, on CD-ROM discs. …
>
> The database in SelectPhone™ cost more than $10 million to compile and is expensive to keep current. It is much more valuable to some users than to others. The combination of names, addresses, and [standard industrial] codes enables manufacturers to compile lists of potential customers. Manufacturers and retailers pay high prices to specialized information intermediaries for such mailing lists; ProCD offers a potentially cheaper alternative. People with nothing to sell could use the database as a substitute for calling long distance information, or as a way to look up old friends who have moved to unknown towns, or just as a electronic substitute for the local phone book. ProCD decided to engage in price discrimination, selling its database to the general public for personal use at a low price (approximately $150 for the set of five discs) while selling information to the trade for a higher price. It has adopted some intermediate strategies too: access to the SelectPhone™ database is available via the America Online service for the price America Online charges to its clients (approximately $3 per hour), but this service has been tailored to be useful only to the general public. . . .
>
> Matthew Zeidenberg bought a consumer package of SelectPhone™ in 1994 from a retail outlet in Madison, Wisconsin, but decided to ignore

[3] 499 U.S. 340, 18 USPQ2d 1275 (1991)
[4] 39 USPQ2d 1161 (1996)

> the license. He formed Silken Mountain Web Services, Inc., to resell the information in the SelectPhone™ database. The corporation makes the database available on the Internet to anyone willing to pay its price – which, needless to say, is less than ProCD charges its commercial customers. Zeidenberg has purchased two additional SelectPhone™ packages, each with an updated version of the database, and made the latest information available over the World Wide Web, for a price, through his corporation. ProCD filed this suit seeking an injunction against further dissemination that exceeds the rights specified in the licenses (identical in each of the three packages Zeidenberg purchased).

The district court[5] found that, in light of the Supreme Court's *Feist* decision that ProCD could not have copyright protection for their telephone directory listings. While ProCD did have a copyright on their computer program used to access the directory listings, Zeidenberg was not using that program to supply listings on his Web site, and therefore did not infringe that copyright. The district court also found that the shrinkwrap license did not meet the requirements for a contract under the Uniform Commercial Code, because the terms were not available to the user at the time of purchase.

But more importantly, the district court found that the shrinkwrap agreement was just a way to get around copyright law, and in particular the uncopyrightability of telephone directory listings, and was therefore preempted by section 301 of the Copyright Act.

The Seventh Circuit was concerned that, without some form of protection for ProCD's directory listings, there would be a "market failure" that would prevent others from creating and marketing databases.

> If ProCD had to recover all of its costs and make a profit by charging a single price – that is, if it could not charge more to commercial users than to the general public – it would have to raise the price substantially over $150. The ensuing reduction in sales would harm consumers who value the information at, say, $200. They get consumer surplus of $50 under the current arrangement but would cease to buy if the price rose substantially. If because of high elasticity of demand in the consumer segment of the market the only way to make a profit turned out to be a price attractive to commercial users alone, then all consumers would lose out – and so would the commercial clients, who would have to pay more for the listings because ProCD could not obtain any contribution toward costs from the consumer market.
>
> To make price discrimination work, however, the seller must be able to control arbitrage. An air carrier sells tickets for less to vacationers than to business travelers, using advance purchase and Saturday-night-stay requirements to distinguish the categories. A producer of movies segments the market by time, releasing first to theaters, then to pay- per-view services, next to the videotape and laser-disc market, and finally to cable and commercial tv. Vendors of computer software have a harder task. Anyone can walk into a retail store and buy a box. Customers do not wear tags saying "commercial user" or "consumer

[5] 38 USPQ2d 1513 (1996)

> user." Anyway, even a commercial-user-detector at the door would not work, because a consumer could buy the software and resell to a commercial user. That arbitrage would break down the price discrimination and drive up the minimum price at which ProCD would sell to anyone.
>
> Instead of tinkering with the product and letting users sort themselves – for example, furnishing current data at a high price that would be attractive only to commercial customers, and two-year-old data at a low price – ProCD turned to the institution of contract. Every box containing its consumer product declares that the software comes with restrictions stated in an enclosed license. This license, which is encoded on the CD-ROM disks as well as printed in the manual, and which appears on a user's screen every time the software runs, limits use of the application program and listings to non-commercial purposes.

It should be noted that much like the Second and Ninth Circuits are noted for their copyright opinions, the Seventh Circuit has a strong "law-and-economics" bent. But how can telephone listings be protected when the Supreme Court has found that they are not protectable by copyright and federal copyright law preempts other forms of protection? Finding that shrinkwrap licenses are not preempted by the Copyright Act.

> But are rights created by contract "equivalent to any of the exclusive rights within the general scope of copyright"? Three courts of appeals have answered "no." The district court disagreed with these decisions, but we think them sound. Rights "equivalent to any of the exclusive rights within the general scope of copyright" are rights established *by law* – rights that restrict the options of persons who are strangers to the author. Copyright law forbids duplication, public performance, and so on, unless the person wishing to copy or perform the work gets permission; silence means a ban on copying. A copyright is a right against the world. Contracts, by contrast, generally affect only their parties; strangers may do as they please, so contracts do not create "exclusive rights." Someone who found a copy of SelectPhone™ on the street would not be affected by the shrinkwrap license – though the federal copyright laws of their own force would limit the finder's ability to copy or transmit the application program.

The Seventh Circuit appears to be holding that any contract, simply because it *is* a contract, will preempt the Copyright Act, with its balance between owner's rights and permitted uses. But such a view would allow a book publisher to override the first sale rule for a book by including a notice that the book can only be read by the purchaser as part of a shrinkwrap license.

> Think for a moment about trade secrets. One common trade secret is a customer list. After *Feist,* a simple alphabetical list of a firm's customers, with address and telephone numbers, could not be protected by copyright. Yet *Kewanee Oil Co. v. Bicron Corp.,* holds that contracts about trade secrets may be enforced – precisely because they do not affect strangers' ability to discover and use the information independently. If the amendment of Section 301(a) in 1976 overruled

> *Kewanee* and abolished consensual protection of those trade secrets that cannot be copyrighted, no one has noticed – though abolition is a logical consequence of the district court's approach. Think, too, about everyday transactions in intellectual property. A customer visits a video store and rents a copy of *Night of the Lepus*. The customer's contract with the store limits use of the tape to home viewing and requires its return in two days. May the customer keep the tape, on the ground that Section 301(a) makes the promise unenforceable?

Are these really the logical consequences of the view that copyright law preempts the attempt to protect the material by a license, or are they red herrings? *Kewanee Oil v. Bicron* was decided in 1974, at the time when publication was a requirement for federal copyright protection. A trade secret would clearly be outside the scope of copyright. In the video store example, the video store is not the owner of the copyright in the tape, and so is clearly not exercising the same rights as provided by the copyright laws. It simply wants its tape back.

The case was not appealed to the Supreme Court, so there has been no opportunity to resolve the differences between the Fifth and Seventh Circuits. But the recent trend is for courts to uphold licenses, including shrinkwrap licenses, that limit the rights of a user under copyright law.

IV. To legitimize shrinkwrap licenses

The *Feist* decision reinvigorated those who want to legitimize shrinkwrap licenses, since they saw that there would be no other way at the time to protect databases with little originality in their selection or arrangement, like directories, or based on material in the public domain, like collections of court cases. (Database providers have also tried for a special federal law protecting databases, but have been unsuccessful.) Even before the *ProCD* decision, they revived the idea of state laws legitimizing shrinkwrap licenses, and having those shrinkwrap licenses limit the use of digital information.

On February 2, 1996, a proposed chapter 2B for the Uniform Commercial Code was published by the National Conference of Commissioners on Uniform State Laws (NCCUSL), the group with representatives from all the states that propose laws governing nationwide issues, to be adopted by state legislatures. It would cover licenses, much as chapter 2 covers sales and chapter 2A covers leases, specifically recognizing shrinkwrap licenses.

There were many objections to this proposal, and also to the revisions that followed. Both major professional societies for computer scientists – IEEE and ACM – wrote to the drafters indicating their concerns, as did consumer groups and even the Federal Trade Commission. In an effort to get the proposal passed, it was decided not to make it a part of the Uniform Commercial Code (which required the approval of a second group, the American Law Institute, that had previously voted against the proposal) and propose it as a new law separate from the Uniform Commercial Code – the Uniform Computer Information Transactions Act (UCITA). On July 29, 1999, NCCUSL passed UCITA and recommended its adoption by the states.

Virginia was the first state to adopt UCITA, with Maryland following close behind. (Actually, Maryland's law went into effect first, because Virginia's had a period for further study amendment before it became effective.) Iowa, after considering UCITA, adopted a bill that nullified any contract that "is to be interpreted pursuant to the laws of a state that has enacted" UCITA and instructed Iowa courts to apply Iowa law to any contract if any party to that contract resides in Iowa or has a principal place of business there. North Carolina, Vermont, and West Virginia have adopted similar "bomb shelter" legislation to protect their citizens from UCITA, and other states have considered such legislation. Faced with this opposition and being unable to get beyond the original two states that adopted UCITA, it is no longer being actively promoted.

IV.A. Problems with UCITA

There were a number of drafts and revisions to UCITA, as the backers tried to apply band-aids to the language in response to criticisms. But UCITA retains the recognition of shrinkwrap licenses and other licenses accepted by action.

Perhaps the biggest problem with UCITA is its size and complexity. Although it is aimed at setting the rules for shrink- and click-wrap licenses with users of computer software and information, it is essentially incomprehensible to a layman. Even a special committee of the American Bar Association politely called it "extremely difficult to understand."

But the problems that critics of UCITA point out are not so much in its language, but in the fact that it legitimates almost any provision that somebody wants to include in a mass market license. The drafters of UCITA did include a limit on terms that could be included, but it covers only the most extreme contract terms. In Section 105(b), they state:

> If a term of a contract violates a fundamental public policy, the court may refuse to enforce the contract, may enforce the remainder of the contract without the impermissible term, or so limit the application of the impermissible term as to avoid any result contrary to public policy, in each case, to the extent that the interest in enforcement is clearly outweighed by a public policy against enforcement of the term.

Under existing contract law principles, a court will refuse to enforce a contract term that goes against any public policy. Whether the modifier "fundamental" to public policy makes a difference or not is yet to be seen, but has raised some concern. The UCITA drafters, in a comment on this provision, stated that "Fundamental state policies are most commonly stated by the legislature. In the absence of a legislative declaration of a particular policy, courts should be reluctant to override a contract term." Does it mean that some things that offend public policy, as long as it is not "fundamental public policy," can be included in a license and be enforceable in court?

UCITA contains a number of provisions that alter the usual copyright terms. By converting what seems to be a sale of a software product into a "license," UCITA allows a vendor to escape two important copyright provisions. Section 109, first sale, only applies to "the owner of a particular copy or phonorecord lawfully made" and Section 117, the special rules for computer programs, only exempt the acts of "the owner of a copy of a computer program." Even though you thought you were buying that computer program or digital information, if the mass market license says you aren't, and UCITA says that term is enforceable (because it's unlikely to be against "fundamental public policy"), then you aren't an owner and don't come under Sections 109 or 117.

Critics of UCITA have pointed out other terms that have been included in past shrinkwrap licenses may not violate a fundamental public policy, and yet are very troubling:

- A prohibition against public criticism or commenting on the software, and in particular the publishing of benchmarks in trade publications.
- A prohibition against using a common web site construction tool in a way that "disparages" the software vendor or its products and services, or promotes "hatred."
- Provisions that prohibit the reverse engineering of a computer program, even if such reverse engineering is found to be a fair use under copyright law.
- The disclaiming of any and all warranties for a computer program, even if there are critical problems with the program known to the vendor but not disclosed to the buyer.
- Choice of law and forum provisions that require the user to bring any action in a distant state.

There are no court cases interpreting UCITA's provisions and whether they are preempted by federal copyright law. If one follows the reasoning of the Fifth Circuit, in *Vault v. Quaid*, then UCITA is preempted where it conflicts with the federal copyright scheme; if one follows the Seventh Circuit's *ProCD* reasoning about the additional element present in all contracts, then it isn't.

V. Using licenses to affect others' actions

Licenses have generally been used to limit the warranties on software, or restrict where it can be run to particular people or machines. But recently there has been interest in a different type of license, one that is used to affect how others license their own software. Although this type of license has been around since the late 1980's, the recent popularity of the Linux operating system, which is distributed under a license of this type, has caused a new interest in it.

The license terms flow from Richard Stallman's interest in having software which is "freely" distributed rather than tightly-controlled by its developer. In particular, he wanted not only the executable software distributed, but also the source code so that other programmers could learn from, fix, and enhance the software. Because he had developed software that was of interest to the computer science community – the Emacs editor and the GCC compiler – he had something to start the ball rolling. He distributed these programs under his "GNU General Public License," which required that any modifications made to the programs covered by the license had to be distributed under the terms of the license.

The GPL is not particularly easy to understand. One commentator has noted that using a common measure for writing complexity, that its basic permission-granting terms receive a score "higher than the most complex major EULAs" (End User License Agreements) and that "To be likely to understand these terms, a reader would require more than 23 years of formal education."[6]

[6] Douglas E. Phillips, *The Software License Unveiled*, Oxford University Press 2009, p. 137.

Unlike proprietary software licenses, this license does not claim that you do not own your copy of the program. “Activities other than copying, distribution and modification are not covered by this License; they are outside its scope.” So the user rights provided by Sections 109 and 117 of the Copyright Act should be available.

To effect the idea behind the license, there were the following provisions.

> You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License. . . .
>
> These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
>
> Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

Some have referred to this as the “virus” provision, since the requirement of source code distribution without a license fee flows to any program derived from a program that is under this license.

While many people conflate “free” software with “open source” software, Stallman clearly doesn’t. In a recent article,[7] he writes:

> Nearly all open source software is free software; the two terms describe almost the same category of software. But they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement. For the free software movement, free software is an ethical imperative, because only free software respects the users' freedom. By contrast, the philosophy of open source considers issues in terms of how to make software "better"—in a practical sense only. It says that non-free software is a suboptimal solution. For the free software movement, however, non-free software is a social problem, and moving to free software is the solution.

Previously in the license, a “work based on the Program” was defined as “either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language.” So this license also addresses the exclusive right to prepare derivative works.

[7] Richard Stallman, “Viewpoint: Why ‘Open Source’ Misses the Point of Free Software,” *Communications of the ACM*, Vo. 52, No. 6 (June 2009), pp.31-33.

> You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
>
> a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
>
> b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; . . .

This is the source code distribution requirement, with simply including the source code in any distribution being the easiest way of doing it. Note that this provision predates the Web, so it addresses distribution on the Web only as a case of subsection b.

> You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

It's not clear whether the "copy" part of the license refers to the copying of the program into RAM for its execution. The license agreement predates *MAI v. Peak*. But if the license pertains only to the copying done as part of a redistribution, then the full Section 117 rights are available to the owner of a copy of the program, including the right to make any copies needed to use the program and to modify the program as necessary for its use.

> You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

Stallman was particularly concerned about software patents, which he strongly opposes.

> If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not

> permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.
>
> If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
>
> It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
>
> This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

The terms of these licenses have not been tested in court, and it is not clear how they might be treated. It is likely the requirement of making one's own source code based on publicly-distributed source code also publicly-distributed will not be a problem. However, if a license were to require the use of another product, such as MS-DOS with Windows, there could be antitrust problems with the tying of the products.

V.A. GPLv3

The current version of the GPL, version 2, was drafted in 1991, two years after the first version. In early 2006, the Free Software Foundation released its first draft of version 3 of the GPL for public comment, and adopted the final version on June 29, 2007. As it noted:

> Many provisions of the GPL could benefit from modification to fit today's circumstances and to reflect what we have learned from experience with version 2. Given the scale of revision it seems proper to approach the work through public discussion in a transparent and accessible manner.

It is envisioned as more than a simple update of the license, however. In a background paper,[8] the FSF characterized the new license as:

- A Worldwide Copyright License
- The Code of Conduct for Free Software Distributors
- The Constitution of the Free Software Movement
- The Literary Work of Richard M. Stallman

The latter presumably means that Stallman had the final say in what the license says, regardless of the input received, and that any changes to the license would

[8] http://www.fsf.org/licensing/essays/gpl3-background.html.

infringe his copyright. This is in line with the two previous versions, which started by stating:

> Copyright (C) 1989, 1991 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Version 3 continues to ignore the user rights provided in the copyright statutes, particularly Section 117 and first sale under Section 109 when it states “nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License.”

Version 3 makes a number of changes that reflects Stallman’s positions. First, it attacks digital rights management for copyrighted works.

> No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

It is questionable whether just saying that something won’t “be deemed part of an effective technological measure” makes it so. If that were the case, the license could say that the software is not a “process, machine, manufacture, or composition of matter” and exempt itself from the patent laws. It’s much like the question asked by Abraham Lincoln: “If you call a dog’s tail a leg, how many legs does a dog have?”[9] But the FSF does not have the final say in this – it will be up to Congress and the courts interpreting the statutes to determine whether a license can exempt itself from a statute such as the DMCA.

Version 3 continues the attack on software patents, but this time trying to address a loophole demonstrated by an agreement between Novell and Microsoft. Version 2 of the GPL required that if you were authorized to distribute a program under a patent license, that patent license has to be available at no cost to everybody distributing or using the program.

> If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

But Microsoft did not give Novell a patent license. Instead, it promised not to use any user who had received their copy of Linux directly from Novell. Novell has no patent license from Microsoft. Instead, Novell users who were not bound by the GPL if they were just using Linux had a promise that Microsoft would not sue them for patent infringement.

This is obviously trying to exploit a loophole in GPLv2, but remember that any ambiguities in the GPL are resolved against its authors, since they drafted its language and offered it on a “take it or leave it basis.”

[9] Four. Just calling the dog’s tail a leg doesn’t make it one.

GPLv3 attempts to close this loophole by defining a “patent license” broadly as “any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement).” And then it tries to make sure that Microsoft-Novell-type agreements will never happen again.

> A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

The final date excludes the agreement between Microsoft and Novell, which was announced on November 2, 2006. The March 28, 2007, date was when the specific approach to patent licenses quoted above was first publicly released.

There was another problem with GPLv2 that GPLv3 tries to address, brought on by TiVo’s use of the GPL-licensed Linux operating system. As is required under Linux’s GPLv2 license, TiVo supplies the source code for the operating system and utilities and any changes it has made. (It does not supply source code for the TiVo applications program.) Everything is there to modify and compile the TiVo-supplied code. But the TiVo hardware included a security mechanism that prevented code not authorized by TiVo to run. (Stallman refers to this is “Tivoization,” and it irks him as much as software patents.)

Version 3 of the GPL adds the requirement that “installation information” be furnished along with the source code.

> “Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made. …
>
> Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

A number of influential software developers, including Linus Torvalds, have expressed opposition to version 3 and have indicated that they plan on continuing using version 2 unless substantial changes are made to address their concerns.

VI. The need for strong IP protection

One might expect that “open source” software would be a good fit for a country without strong intellectual property laws. After all, the software developers are making the source code for their programs available to the public at no cost, and allowing its further redistribution.

However, there are substantial differences between “open source” or “free” software and software that is actually in the public domain, without any form of intellectual property protection. Both “open source” and “free” software depend on licenses, and the threat of legal action based on copyright infringement or breech of contract, to impose requirements on downstream distributions of the software and its modifications. That requires strong intellectual property and contract laws for those requirements to be more than just nice ideas.

But those same strong intellectual property laws can also hamper the reimplementation of a proprietary program as open source. Any country wishing to create an environment for open source development needs to be very careful in structuring its intellectual property laws.

VI.A. Open Source and Downstream Requirements

The ideals of both “open source” and “free” software permit anybody to modify and redistribute that software, but that is substantially different than placing the software in the “public domain.” Rather than impose no restrictions, as would be the case if the software were really in the public domain, open source software licenses generally require that any redistribution of the modified software be under the same terms as the original license.

The Open Source Definition[10] covers this in Section 3, “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.
>
> Rationale: The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

This represents the *quid pro quo* for open source software – that you may use it as the basis for your implementation or modifications, but those modifications must be available to the community on the same terms as the original software. A company cannot take the work of an open source developer, include it in a program, and distribute it as a proprietary program whose source code is not provided or without the downstream right to modify and redistribute.[11]

[10] http://www.opensource.org/docs/definition.php

[11] In the religious debates over “open source,” and particularly “free” software, sometimes this inclusion of another developer’s software into a proprietary program gets referred to as “making the open source software proprietary.” But that is not what happens. The original open source program is still available on the same terms as it originally was. The only thing that may be unavailable is the specific proprietary modifications that were made.

VI.B. The Need for Strong IP Laws

For the open source rules to have any meaning, there must be both strong copyright and contract law. Without a strong copyright law that gives the owner of the copyright of a work the right to control the distribution of derivative works, there is not need for an open source license to permit such a distribution. One does not need the permission of a license to do what one is already permitted to do. Without strong contract law, the special terms imposed by an open source license on any distribution are not easily enforced.

As an example of the problems that weak intellectual property laws can cause for open source licenses, consider the effort NASA has made to develop the "NASA Open Source Agreement."[12] Under United States copyright law, software produced by NASA employees cannot be protected by copyright.[13] Instead, it attempts to use contract law to impose obligations on the recipient of the software similar to those of a conventional open source license.

But while somebody who violates the terms of a conventional open source license can be sued for copyright infringement, that is not the case with the NASA license, since NASA owns no copyright in the software. At best, the violator can be sued for breech of contract. And if the software was obtained outside of the license, such as through a Freedom of Information Act request or perhaps from someone in violation of the license, there is no license between NASA and the party obtaining the software, so the party is free ignore the terms of the NASA license.

The GPL explicitly recognizes this need for both strong copyright and contract laws to achieve its goals:

> Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

VI.C. Similar to the Needs for Proprietary Software

But strong copyright protection and the recognition in contract law of licenses that accompany software are also what protects proprietary software. Both want to have the terms of their license enforceable under the law, and to use copyright law to cover people who may not be party to the license or as the reason that a license is necessary. Both use the license to disclaim as many warranties as possible.

Of course, there are differences in the license terms between open source and proprietary software. Proprietary software licenses are generally concerned with the way that the software may be used (on a single machine, limited number of backups, no reverse engineering, etc.) while open source licenses are generally concerned with the modification and distribution of the software. But both attempt to control user behavior by license terms in trade for permitting what would otherwise be a copyright

[12] http://opensource.arc.nasa.gov/pdf/NASA_Open_Source_Agreement_1.3.txt

[13] "Copyright protection under this title is not available for any work of the United States Government, but the United States Government is not precluded from receiving and holding copyrights transferred to it by assignment, bequest, or otherwise." 17 U.S.C. §105.

infringement. (Distribution in the case of open source; reproduction necessary to use the software in the case of proprietary programs.[14])

It may seem strange to talk about how the legal structures needed for open source also provide the foundation for trade secret protection, because open source is the antithesis of trade secrets. But the foundation for trade secret protection – other than never disclosing the software to anybody – is contracts that govern what a person receiving a trade secret may do with that information. And the strong contract law that is necessary for open source is the same law that supports trade secret agreements.

VI.D. But Not Too Strong

But one has to be careful not to have copyright laws that are too strong. Many open source programs are based on existing proprietary programs. The Free Software Foundation's GNU project is trying to create a "free" version of the Unix operating system, which was proprietary to Bell Labs at the time the GNU project started, and was only available under a signed license with terms too restrictive for the Free Software Foundation's founder. (The name GNU is a recursive acronym "GNU is Not Unix," although it was trying to be.) Linux (or GNU/Linux, as the Free Software Foundation would like it) is the kernel that the GNU project seemed unable to get in place, and when combined with the other GNU programs gives an excellent reimplementation of Unix.

There are two areas where open source development of programs to replace existing proprietary programs can run into difficulties: the copying of the nonliteral aspects of the existing program, and when a new program is a derivative work of an existing program.

In the United States, the abstraction-filtration-comparison test is used to determine which nonliteral aspects of a computer program are protected and whether there has been an infringement. The test is not well-specified and requires a considerable effort to apply, but there are a number of reasons why an element of a program is not protectable by copyright:

- The element's expression was dictated by reasons of efficiency, such as when it is the best way of performing a particular function.
- The element's expression was dictated by external factors, such as using an existing file format or interoperating with another program.

[14] In this respect, the proprietary software companies are also trying to use their "license" as an end run around 17 U.S.C. 117, which permits the "owner of a copy of a computer program" to make the incidental copies necessary to run a computer program. The license claims to make the purchaser of the software not its owner, but merely a licensee. The CONTU Final Report had originally suggested granting the right to any "rightful possessor of a copy" but Congress, with no explanation, substituted the current language. In *MAI v. Peak* (991 F.2d 511, 26 USPQ2d 1458, CA9 1993), the Ninth Circuit noted that licensees "do not qualify as 'owners' of the software and are not eligible for protection under Section 117." One possible reason is that Congress was concerned about extending this right to renters of computer software, who could legally make copies and then conveniently forget to delete them, but changes to 17 U.S.C. 109 in 1990 eliminated that reason.

- The element's expression is a conventional way of writing something in the particular programming language or machine running the program.
- The element, at the particular level of abstraction, is an unprotectable process and not protectable expression.
- The element is taken from the public domain or is an unprotectable fact.

Copyright laws that do not provide an exception for expression dictated by external factors, for example, can make it impossible to create a new work which is based on a preexisting work, particularly if it is necessary to be compatible with file formats, as would be the case for an open source word processor that needs to be able to read existing Microsoft Word documents for people to adopt it.

VI.E. What About Patents?

Probably the most contentious area of intellectual property when it comes to open source is patents. The Free Software Foundation is adamantly against patents on software-based inventions, as indicated in the GPL Preamble:

> Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

This position should not be surprising, since the original goal of the Free Software Foundation's GNU project was to create a "free" version of the Unix operating system. While it is possible to use techniques such as "clean rooms" to assure that the copyright of an existing program is not violated when it is reimplemented by somebody, there is no similar way to protect infringing a patent that may cover some critical aspect of a program. Independent development is not a defense to patent infringement.

This is more than a hypothetical argument. United States Patent 4,135,240, issued in 1973, covers a key aspect of the Unix operating system – the ability to set the identity of the effective user for a program based on control information in the program's directory entry. This involves an additional bit (called SUID) that indicates that the effective user should be the owner of the program file, not the current user. This allows users to run selected programs as if they were a system administrator, and is used for many important Unix functions. It would be impossible to implement an operating system fully-compatible with Unix without infringing this patent.

Luckily for the GNU project, the patent owner (Bell Labs) dedicated the patent to the public shortly after it had issued,[15] but the lesson was clear. Just one patent on a key aspect of a program could prevent the development of a "free" version of that program.

The Open Source Definition takes no position on patents.

To the extent that open source software remains primarily the reimplementation of existing proprietary software, patents will remain a real problem, and one even more

[15] It presumably was one of the many patent applications that Bell Labs had filed to test the boundaries of software patentability.

critical since software companies like Microsoft have discovered software patents and have been building their portfolios.[16] These patents can be easy to get if one doesn't want breadth in coverage, but instead will be content with a patent that covers only a specific feature so that a program cannot be exactly reimplemented. One Microsoft patent (6,286,013) is so narrow that some of its claims recite the specific function codes used by Microsoft DOS or Windows. But it would be impossible to write an operating system capable of running Windows programs without infringing this patent.

But as open source developers move from reimplementing proprietary programs to developing new programs and techniques, patents may be the only way to prevent proprietary software developers from free-riding by reading the open source code to learn the techniques, and then doing their own implementation that does not infringe the copyright on the open source program.

Many companies also have extensive patent portfolios so that when they seem to be blocked by a patent, they have something that might be cross-licensed to get permission to use the blocking patent. If open source developers want to fight patents that may block their activities, they may want to develop patent portfolios of their own, and perhaps pool their patents toward the common good.

VI.F. Going too far?

Can the requirements imposed by a license go too far? In *Lasercomb America v. Reynolds*,[77] the Fourth Circuit said yes.

As part of the Lasercomb agreement, a licensee had to agree not to develop a competitive computer-aided design program for 99 years, well beyond the period of protection given Lasercomb's program by the copyright laws at that time, 75 years. The court found that Lasercomb was trying to effectively extend the term and scope of its copyright beyond what copyright law permitted, and that would prevent people from legitimately developing competitive software. That was a misuse by Lasercomb and the court refused to enforce their copyright against Reynolds.

The copyright misuse doctrine is similar to the better-developed patent misuse doctrine. The classic patent misuse occurs when a patent owner conditions the use of a patented item (such as a salt spreader) on the purchase of a nonpatented item (such as salt) also supplied by the patent owner. The courts have found that such an action attempts to improperly enlarge the scope of the patent, and therefore when the patent owner comes to court, it is with "unclean hands" and the court will refuse to enforce the patent until the misuse ends and its effects no longer exist. The misuse does not have to be against the alleged infringer – any misuse can be used to defend against the infringement suit.

The *Lasercomb* case has been cited with approval in a number of cases since it was handed down, but in most of those cases copyright misuse was not found. While the exact dimensions of the copyright misuse defense will be known only after considerably more cases are decided, its consequences should be considered by anyone who is trying to use his or her copyright to go beyond the protection of the copyright laws. The penalty for copyright misuse – unenforceability of the copyright in

[16] As of the beginning of May 2004, Microsoft held over three thousand patents.
[77] 911 F.2d 970, 15 USPQ2d 1846 (4th Cir. 1990).

court until the misuse has been purged and its effects no longer exist – is tantamount to losing the copyright.

VII. Additional Reading

Recently, two good books have come out that discuss open source licensing:

- Open Source Licensing: Software Freedom and Intellectual Property Law by Lawrence Rosen, general counsel of the Open Source Initiative. (Prentice Hall, 2005, ISBN 0-13-148787-6)
- *Understanding Open Source & Free Software Licensing* by Andrew St. Laurent. (O'Reilly, 2004, ISBN 0-596-00581-4)

In addition, information about free and open source licensing, including a variety of current licenses, can be found on the web pages for:

- the Open Source Initiative (http://www.opensource.org/)
- the Free Software Foundation (http://www.fsf.org/)
- GPL version 3 (http://gplv3.fsf.org/)

The leading decision upholding shrink-wrap licenses is *ProCD v. Zeidenberg*, decided in 1996 by the Seventh Circuit.

Information about UCITA can be found from:

- NCCUSL, the National Conference of Commissioners on Uniform State Laws, which is the group that prepared UCITA http://www.nccusl.org/)
- AFFECT, a group opposed to UCITA (http://www.ucita.com/)