10 Tex. Intell. Prop. L.J. 349

Texas Intellectual Property Law Journal
Spring 2002

Article

OPEN SOURCE LICENSING: VIRUS OR VIRTUE?

Christian H. Nadan[a1]

Table of Contents

Although open source licensing is more than fifteen years old, it has only recently gained popular fame as well as notoriety. It has been hailed as a panacea for the ills of the technology industry, and damned as a frontal attack on free enterprise. Regardless, its proponents and critics would both agree that open source licensing--free sharing of computer software with a community of independent programmers who collaborate on the software's development--has become an important part of the current technology industry. In the hands of a legally savvy software developer, open source licensing can be a valuable tool. Indeed, commercial enterprises are working with, incorporating and even distributing open source software, as it moves further away from its noncommercial, academic roots.

This Article examines open source licensing, its legal implications, and most critically, the use of open source licensing in the commercial context. Without a thorough understanding of open source code and licensing, those using or distributing open source code may be in for a surprise--even losing their ***350** commercial rights to their own software products. In explaining and analyzing open source licensing, this Article can also guide the practitioner wishing to add open source licensing to his or her arsenal of skills to be deployed in representing and advising technology companies.

Part I of this Article explains open source licensing--what it is and how it developed, its philosophical underpinnings, and the standard open source license agreements. This Part also analyzes the legal implications of open source licensing, including

the drastic consequences to commercial enterprises that fail to employ open source licensing correctly. Understanding these risks is essential to work effectively with open source licenses and open source software. Part II explores some of the legal uncertainties in open source licensing, and failures in the standard open source license model that may undermine even a correct open source licensing program. How commercial enterprises react to open source licensing is the topic of Part III. This final part first discusses Microsoft's response to open source licensing, and concludes by proposing a set of principles and business models for using open source licensing effectively in the commercial context--to make it serve you instead of the other way around.

I. Open Source Licensing

A. A Basic Understanding of Computer Software

To understand open source licensing, one must understand the basics of computer software. Software programmers write software programs using a high level computer language such as BASIC, C++, or Java. These high level languages generally use English alphanumeric characters and symbols and enable the developer to tell the computer what to do. For instance, the command "'GO TO 40' tells the computer to skip intervening steps and go to the step at line 40" of the program, which in turn might instruct the computer to add two numbers.[1] A computer program written in this high level language is in "source code" form.[2]

As one might guess, a computer does not read alphanumeric instructions. Rather, the computer responds to binary inputs, essentially "using two symbols, 0 and 1, to indicate an open or closed switch (e.g., '01101001' means, to the Apple [computer], add two numbers and save the result)."[3] Thus, after writing the program in source code, the programmer must use a compiler program to translate or "compile" the source code into the corresponding 1s and 0s that the computer ***351** can read. Source code compiled into this series of 1s and 0s is called "object code."[4]

Computer programs are typically distributed in object code form. Thus, when you go to Best Buy, OfficeMax or your local office equipment supply store to buy a copy of Microsoft Office software, it comes on a CD-ROM into which the object code 1s and 0s have been encoded. This makes sense. The typical user wants to take the software to her computer, install it and have it run; she does not want to have to buy source code, obtain the proper compiler software to go with it, and then try to compile the source code into a proper object code program before being able to use it.

The software developer shares the same interest in distributing the software in object code form. Not only does the software developer want its software to be convenient to use, it wants to protect its source code from disclosure. Because humans can read source code, an engineer can examine source code to find out how a program works, and see the embodiment of the original programmer's skill, effort, creativity, and innovation. If software were distributed in source code form, any skilled engineer could take and reuse the innovative or labor-intensive parts of the program in that engineer's own competing program. This would eliminate the competitive advantage of the first program, since the second program would have the same innovative features, or would at least be much cheaper to develop (since the first programmer did all the hard work already) and thus could be priced more cheaply. This appropriation of the original programmer's work may violate intellectual property laws, but is extremely difficult to discover, and requires expensive litigation to stop. Distributing software in object code form is a more cost-efficient and effective means to prevent such misappropriation.

In addition to these competitive reasons, there are also practical reasons not to disclose the source code. With access to source code one can change the original program by modifying the alphanumeric instructions and then compiling it to create an enhanced or altered version of the original program. The original programmer may oppose this conduct on the grounds it will prevent the original version of the program from becoming a uniform standard, or because it will be more difficult to provide support to end-users if they are modifying the program.[5] Finally, a software developer can obtain trade secret protection for the source code, as long as the source code is secret and she has taken reasonable steps to maintain its secrecy.[6] One important way of keeping the source code secret is to distribute only object ***352** code copies of the program. For all these reasons, the primary means for distributing software is in object code form only.[7] This is often referred to as the proprietary software model.

B. What is Open Source Licensing?

Open source licensing refers to the practice of making the source code for a software program openly available at no charge for the public to work with it. Typically, a community of developers works with the source code, creating enhancements and extensions and improvements, which are in turn freely shared back with the community and further enhanced, extended and improved. With proprietary software, enhancements and bug fixes are dependent upon the schedule and employees of a single company. Conversely, with open source software, an entire community of developers around the world is available to provide enhancements and bug fixes, and if you are not satisfied with their pace or performance, you can simply do it yourself.

Usually, the original software developer or a small group of interested programmers will act as a de facto project manager, controlling what new code is incorporated into the evolving software program, and ensuring that any new enhancements, extensions, or improvements are suitably well written to be checked into the official code base. The project manager "may act as an official arbiter of versions, and periodically release official improved versions of the original source code to incorporate other programmers' modifications[.]"[8] The now well known GNU Linux operating system has evolved in just this way. It is an open source operating system "that was created, and is continuously updated, by a global network of software developers who contribute their labor for free."[9] In the open source community's own words,

The basic idea behind open source is very simple: When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.[10]

Within hours after the Netscape browser's source code was released as open source, a team of Australian programmers had contributed to the code a feature that ***353** enables secure Internet transactions.[11] In less than a month, the open source community had finished a new version of the browser.[12]

C. The Free/Open Source Movement

As explained below, the open source community divides itself into "open source" and "free software" camps. Essentially, their concept of open source licensing is the same, but their motivations for open source licensing differ. For purposes of this Article, I refer to open source (without quotes) as the generic term incorporating both camps.

The open source movement traces its origins to the "free software" movement. A reaction to the proprietary software model, the free software movement was founded in 1984 on the ideals from 1776--freedom, community and voluntary cooperation.[13] Since source code is needed to modify software, and since proprietary software companies distributed only object code copies of their software, the end users of the software were entirely dependent on the software company to provide bug fixes and upgrades. The end user had to wait for the company to provide these, and to pay whatever price the company imposed. Frustrated with this proprietary software model, the free software movement rested on the principle that the public should have "the freedom to study, change and redistribute the software" it uses or obtains; "these freedoms permit citizens to help themselves and help each other, and thus participate in a community."[14] As its leading voice, Richard Stallman put it, "the 'free' in 'free software' refers to freedom, not price [.]"[15] He notes that this "contrasted with the more common proprietary software, which keeps users helpless and divided: the inner workings are secret."[16]

***354** The free software community thus began work on the free software open source GNU operating system, an effort they called the GNU Project.[17] Over time, the developer community developed, evolved, and finished more and more of the components of a complete operating system. Once Linus Torvalds added the Linux kernel, this combined GNU/Linux operating system was complete, and is what we generally call the Linux operating system today.[18] Many other smaller open source projects were completed as well, providing much of the software underlying the Internet.[19]

The "open"--as opposed to "free"--source movement arose in reaction to Netscape's announcement that it planned to give away the source code of its browser software in early 1998.[20] A group of software developers got together because "[w]e realized that the Netscape announcement had created a precious window of time within which we might finally be able to get the corporate world to listen to what we have to teach about the superiority of an open development process."[21] Rather than following the free software movement, which argued that software should never be proprietary (attacking the proprietary model upon which all commercial software companies were founded), the "open source" movement sought to co-opt the business world: "We realized it was time to dump the confrontational attitude that has been associated with 'free software' in the past and sell the idea strictly on . . . pragmatic, business-case grounds[.]"[22] They called this new movement "open source,"

to distinguish it from the anti-business "free software" movement.[23] The label "free software" troubled them in any event, because it could be confusing whether "free" referred to price, dedication to the ***355** public domain, or distribution under an open source license.[24] The open source movement was eventually organized into an entity called the Open Source Initiative ("OSI").[25]

OSI takes a more pragmatic approach than the free software movement. Both want open source licensing to become more widespread, but instead of positioning it as a threatening replacement for the commercial software industry, OSI positions the commercial software industry as an ally to help spread the use of open source licensing. OSI argues that open source licensing is useful to the business world. "The foundation of the business case for open-source is high reliability. Open-source software is peer-reviewed software," proofed and tested by a whole community of developers.[26] By leveraging a community of open source developers, a software program can be written more quickly and more cheaply--in effect one can outsource some of the work to the open source community, which does not charge for its services.[27] For a small company, open source also affords an opportunity to quickly gain market share or mind share, if the company can interest the open source community in taking up the project.[28] The next Linux is out there.

D. Open Source Licenses

One of the critical factors in open source licensing is the license itself. The license determines how the open source community will develop and distribute the software, and determines the evolution of the open source project. This Article will focus on two basic licenses representing opposite ends of the spectrum of open source licenses: the GNU General Public License ("GPL"), and the Berkeley Software Distribution ("BSD") license.[29]

***356** OSI has published a definition enumerating the elements required for a software license to qualify as an open source license.[30] An agreement merely licensing source code on a confidential basis is not an open source license, for instance. The OSI open source definition has 9 criteria:

1. The license must permit anyone to further distribute the open source software for free as a component of an aggregate software distribution containing programs from several different sources.

2. The open source software must be made available in source code form and must be freely distributable in source code form. This enables others to gain access to the source code, in contrast to the proprietary software model.

3. The license must allow modifications to the source code, and allow redistribution of the modified software under the same terms as the original open source software.

4. The license may restrict redistribution of modified software, however, as long as it permits distribution of the modifications themselves in source code form. This enables the initial code base to remain in its pristine state, so the community can see what modifications are available or proposed.

5. The license must not discriminate against any person or group.

6. The license may not restrict the use of the code to a particular field of use. Thus, it cannot restrict use of the code for educational purposes only, for instance.

7. The licensed rights must apply to all to whom the program is redistributed without the need for execution of an additional license. This prevents indirect means of making openly available source code less open, such as by requiring a separate non-disclosure agreement.

8. The licensed rights must not be dependent on using the software only as part of a larger work with which it is initially obtained.

9. The license must not restrict other software that is distributed along with the licensed software. For example, the license must not require that all other programs distributed on the same CD-ROM must also be open-source software.[31]

***357** OSI has accepted both the GNU GPL and the BSD License as approved Open Source licenses, despite their polar

positions on the spectrum of open source licenses.[32]

The Free Software movement classifies open source licenses slightly differently from OSI, focusing primarily on five key elements:

1. The freedom to run the program, for any purpose.

2. The freedom to access the source code, and modify it.

3. The freedom to redistribute copies of the program.

4. The freedom to release your modifications to the public.

5. Is it a "copyleft" license?[33]

The four freedoms are consistent with (albeit broader than) the criteria employed by the Open Source Initiative. The fifth element--is it a "copyleft" license?--is what sets this definition apart. Copyleft is not itself a freedom, but a means to ensure that the four freedoms cannot be withdrawn from the public.[34]

The Free Software Foundation explains:

> The simplest way to make a program free is to put it in the public domain, uncopyrighted. This allows people to share the program and their improvements, if they are so minded. But it also allows uncooperative people to convert the program into proprietary software. They can make changes, many or few, and distribute the result as a proprietary product. People who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away. . . . So instead of putting GNU software in the public domain, we 'copyleft' it. Copyleft says that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it. Copyleft guarantees that every user has freedom. . . . Proprietary software developers use copyright to take away the users' freedom; we use copyright to guarantee their freedom. That's why we reverse the name, changing 'copyright' into 'copyleft.'[35]

In other words, copyleft licenses provide that you may distribute the open source code and any modifications to it, but only under the same open source license under which you received it. In this way, as the code and modifications to it pass from person to person or entity to entity, they remain open source. This ***358** ingenious tactic is embodied in the GPL, but not the BSD license. Thus, the Free Software Foundation characterizes the BSD license as free, but not copyleft.[36]

The GPL is a copyleft license. Its Preamble boldly declares that it is "intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users."[37] The GPL provides that you may copy and distribute copies of the program's source code, provided that you conspicuously publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to the GPL and to the absence of any warranty; and give any other recipients of the program a copy of the GPL along with the program.[38] You can charge money for the physical act of transferring a copy.[39] In addition, you can modify the program, and copy and distribute your modifications or the entire modified program under the above terms, provided that you meet three conditions: (i) you must include notices on any modified files; (ii) in most cases, you must cause the program to display a notice that users may redistribute the program under these conditions, and telling them how to get a copy of the GPL; and (iii) "[y]ou 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."[40] This third condition is the copyleft provision, and the most controversial feature of the GPL.

The goal of the copyleft provision is to prevent anyone from taking the open source code private (in other words, distributing it under a proprietary, non-GPL license). Consider the Linux operating system. This robust operating system is the product of the free labor of hundreds of skilled programmers over many years, and it is successful enough to be used in many commercial environments. Without copyleft, a party could obtain the open source code for Linux, make some enhancements, and then license the enhanced operating system under a proprietary model--selling object code copies of this new operating system, but not revealing the source code. This party would have contributed very little to the enhanced program, and would

be appropriating the labor of all those programmers who shared their modifications for free. Such conduct would likely limit the success of open source software, since the free labor open source community would be reluctant to share enhancements if someone else could take them private and not ***359** share his own enhancements. Thus, requiring that any derivative works of GPL code also be covered by the GPL seems reasonable, since but for the GPL license, the user would have no rights to create the derivative works in the first place.

Indeed, the GPL goes on to assure that "it is not the intent of this section to claim rights or contest your rights to work written entirely by you."[41] Thus, the copyleft provision applies 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."[42] The next sentence of the GPL, however, potentially undermines the prior language, and has catastrophic implications for the unwitting user of GPL software: "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."[43] This chilling conclusion is buttressed by the condition to the license itself: "You must cause any work that you distribute or publish, that in whole or in part contains . . . the Program or any part thereof, to be licensed . . . under the terms of this license."[44] Thus, if you incorporate some GPL code in your proprietary software product, arguably your whole proprietary product becomes open source and must be licensed by you under the GPL.[45] This illustrates the importance of fully understanding open source licensing before deploying it in a commercial context.[46]

Imagine if one of the thousands of Microsoft software developers, working on his small piece of the franchise software product Microsoft Windows (a separate and independent work), while surfing the internet came across a few lines of GPL code and included them in his work, which eventually became part of Windows. According to the GPL, Windows would suddenly become an open source program, freely available to all. A startup software company, short of programmers, might for convenience decide to include a small piece of GPL code, perhaps a simple utility, in its software product. Many small software companies have not mastered the terms and conditions of the GPL--and they might lose the ability to market ***360** their own product. Indeed, the OSI explains that GPL programs contaminate proprietary code to which they will be actively linked at runtime.[47] This means that even if the startup software company was savvy enough not to include the GPL code in its product, but provided it as a separate product that only shares data with the proprietary product, the startup may still lose. The OSI interpretation suggests that a completely separate proprietary program will become GPL if it merely shares data with GPL code, even if the only such sharing occurs while the program is actually running.[48] This phenomenon--that proprietary code becomes open source whenever combined with open source code--is the so-called "viral effect" of copyleft licenses. Like a virus, GPL code infects any proprietary code with which it is combined, turning it into GPL code as well.[49]

The GPL also provides that you may copy and distribute a program in object code form, provided that you include a copy of the source code, or make it easily available for no more than the cost of the physical transfer, under the terms of the GPL.[50] Finally, the GPL includes a disclaimer of warranties and a provision absolving the developer of the program and all contributors from any liability.[51]

***361** Thus, the GPL is the prototypical copyleft license, and for the reasons above, must be used very cautiously. The BSD License, on the other hand, is a more genteel open source licensing vehicle. It provides simply that:

> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.[52]

In short, anyone is free to use BSD code, as long as the required notices are included (one of the notices in turn requires that subsequent distributors also include the notice, so that it passes from user to user). But this does not prevent a user from taking the BSD code private. As long as the proprietary software includes the required notices, it can be distributed in object code form, and the source code can be maintained in secrecy.

II. Open Source License Enforceability Issues

This Part of the Article will explore some flaws in the standard open source licensing model. Once the risks outlined above have been addressed, and a decision is made to release software under an open source license, there is still a potential for problems. Even if the open source licensing is done correctly, exactly as prescribed in the open source license itself, the flaws in the licenses may enable downstream licensees to evade copyleft, and take open source code private. These flaws can be addressed and the risks minimized, if the developer or her attorney is aware of them.

*362 A. Contract Formation

Proprietary software is typically licensed under a clickwrap or shrinkwrap license. Such licenses serve as a gate through which the user is forced to pass to use the software, and require some affirmative act manifesting assent to the license--such as clicking an "accept" button before downloading the software or to complete the installation process, or tearing open the packaging displaying the shrinkwrap license.[53] There is no way to avoid the act constituting assent. Even then, there is still some doubt over the enforceability of these types of licenses.[54]

Open source licensing tries to make the license formation simpler for the licensor to implement. Rather than requiring acceptance of a clickwrap agreement for downloads from a website, acceptance of a clickwrap license that pops up during installation of the code, or acceptence of a shrinkwrap license included in the packaging, the open source licenses just require that a notice about the license be provided in or with the code. No manifestation of assent is required to prove the licensee agreed to the terms, and indeed the user can access the code without ever seeing the license, let alone agreeing to it. This greatly increases the risk that no license agreement has been formed.

Specifically, the GPL and the BSD both rely on this rudimentary "notice" form of license. The BSD License requires that redistributions of the program "must reproduce the above copyright notice, this list of conditions," the endorsement prohibition and the warranty and limitation of liability disclaimer.[55] So the BSD license relies simply on a notice of the license terms. The BSD notices might be faithfully reproduced in the middle of the code somewhere, and never seen by the licensee. Similarly, under the GPL you may distribute copies of the program and simply provide the licensee a copy of the GPL with the program. What if the copy of the GPL is properly provided with the code, but placed in a file never opened by the user? In neither of these licenses is the user required to perform any act manifesting assent to the terms or even notice of them.[56] Thus, ***363** particularly because neither the BSD License nor the GPL specify how or where the notice is to be provided, it is possible that a user could obtain such software without notice of the terms, even though the distributor is in full compliance with the GPL or BSD License.[57]

If the user has no notice of the license, she is probably not bound by it. Consider Ticketmaster Corp. v. Tickets.Com, Inc.[58] The Ticketmaster website contained at the bottom of the webpage a "terms of use" link. The Terms of Use page provided that going beyond the home page constituted agreement to the Terms of Use. Unless the customer happened to notice and then click on the "terms of use" link, however, he would never see the actual Terms of Use page. The court ruled that this was not like a shrinkwrap license on a box of software, since a shrinkwrap is "open and obvious and in fact hard to miss."[59] Also, the court noted that this was not like the many web sites that required the user to click on "agree" to the terms of use before proceeding. The court concluded that using a terms of use link at the bottom of the page does not necessarily create a contract. It dismissed the breach of contract claim with leave to amend to show that defendant had knowledge of the terms and at least impliedly agreed to them. Thus, simply including contract terms did not create a contract where the user was unaware of them (and thus had no idea that use of the website constituted acceptance of a contract).

Similarly, in Specht v. Netscape Communications Corp., the court held that no agreement is formed if the customer downloads the software without clicking a clickwrap and without first being informed that the downloading constitutes acceptance of the license agreement.[60] The customer could download this software from Netscape's website without being first forced to click the clickwrap. Further, ***364** the notice about the existence of the clickwrap was not even visible on the screen when downloading the software--the user would only discover the clickwrap by browsing elsewhere on the website.[61] "The only hint that a contract is being formed is one small box of text referring to the license agreement, text that appears below the screen used for downloading and that a user need not even see before obtaining the product."[62] Thus, the downloader was "not bound by inconspicuous contractual provisions of which he was unaware."[63]

This is strikingly similar to open source licensing--relying on a simple notice that the user may not ever see. The user is not

forced to click anything or rip open any marked packaging. This "notice" form of licensing will be effective if the user happens to notice the license, and understands that by using the software he is agreeing to be bound. But it does not guarantee that all users will see the license, or that they will understand that downloading or using the code constitutes acceptance of it.[64] Simply stated, a user is not bound by a contract of which he is not made aware.[65]

***365** If the user is not bound by the license, can he still use the software? The GPL argues that "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."[66] This is overstated, if not incorrect. With respect to the software owned by the distributor, providing it to a user without any express (or at least enforceable) terms or conditions can create an implied license to exploit the work on reasonable terms. "[I]mplication of a license depends upon all the circumstances, including the parties' conduct, the terms of applicable written agreements and correspondence, the reasonable expectations of the parties, the dictates of fairness and equity, and the policies underlying the intellectual property system."[67] In the absence of an agreement, "implied licenses may arise from conduct alone. For example, when a copyright owner provides another with a copy of a copyrighted work, intending it to be used, and knowing that it will be used, for an agreed purpose, the recipient normally has an implied license to use it for that purpose, but no other."[68]

There is no question that the open source code is provided for the express purpose of being modified, used and distributed; and if the licensee is not bound by the open source license, the licensee may have an implied license for that purpose. Indeed, fairness requires this result. If the licensor is making the code available for download and use, and through no fault of the licensee he does not receive notice of the actual open source license, it would be inequitable to find that the licensee infringed by just downloading and using the code as intended.[69] If the license agreement is not binding, then either the licensee has an implied license or is infringing. The latter result would reward a licensor for inducing the licensee to exploit the code and failing to adequately notify the licensee of the written license. ***366** A licensor should not be able to give the licensee the code without notice of the license agreement, and then sue the licensee for using the code.[70]

An implied license would not extend to those portions of the program the distributor received from others, however. The distributor cannot grant more rights than he received, and since he received the contributed portions subject to the GPL limitations, he cannot pass on those portions with fewer restrictions.[71] Moreover, many experienced programmers would be hard-pressed to claim that they did not suspect that the code was subject to the GPL, or that their failure to look for the GPL notice was reasonable. But it does not follow that every programmer who downloads the code should have known it was GPL and not public domain code, or that all open source software includes the GPL notice in a conspicuous place. All it takes is one innocent downloader: if such user is not bound by the GPL or BSD License but receives the code under an implied license, this loophole could undermine the open source nature of the program, or at least the portions contributed by the licensor.

***367** Thus, diligently following the GPL rules may nonetheless fail to protect the code. If a recipient of that GPL code reasonably fails to receive notice of the license terms, she could have an implied right to use and distribute that code. It is unlikely a court would imply a term as unusual as copyleft without any notice of it, so the implied licensee may well be able to combine proprietary code with no tainting, or even take the GPL code private by licensing it out under a proprietary license. One initial solution is to set up the code download site so that the user is forced to click "I accept" to a clickwrap form of the GPL before downloading. Another slightly riskier approach is to provide a prominent notice on the download page that the use of the code is subject to the GPL (and provide a copy of GPL on the site or indicate where a copy can be found in the code). Commercial entities rely on clickwraps and shrinkwraps every day, and employing the same device for the GPL is the prudent approach. Indeed, most commercial software distributors already have such mechanisms in place for their proprietary code, so distributing GPL code similarly should be simple and inexpensive. Relying on a mere notice in the code, while authorized in the GPL, is not recommended.

This does not resolve the problem, however. The clickwrap GPL would bind the distributor's licensee. Yet that licensee could still comply with the GPL and redistribute the code with only an unseen nonbinding notice of the GPL. Thus, the GPL would not bind that licensee's licensee. An effective solution must bind the licensee's licensee (and subsequent downstream licensees) to use an enforceable contract formation mechanism for their distributions. Ideally, the GPL should be modified to expressly require that all downstream licensees receive effective notice that the GPL binds them.[72]

B. Enforceability of Copyleft

The copyleft provision of the open source license itself creates a possible unanticipated enforceability problem. This section discusses some of the risks of enforcing or relying on copyleft provisions. To use copyleft licenses most effectively, the developer should be aware of these risks and the limits on copyleft's effectiveness.

Even if the open source license is binding, the copyleft provision may still not be enforceable as to independent proprietary code, in light of the intellectual property misuse doctrine. The doctrine is asserted as an affirmative defense to an intellectual property infringement claim. Much like an unclean hands defense, the misuse doctrine precludes enforcement of intellectual property rights that have been extended beyond the scope of those rights. An obvious example is a contract that requires a patent licensee to continue to pay royalties beyond the statutory term of ***368** the patent.[73] A patent generally lasts for 20 years from its filing date.[74] Demanding royalties after the patent expired would exceed the scope of the rights granted in a patent. Public policy "forbids the use of the patent to secure an exclusive right or limited monopoly not granted by the Patent Office."[75] A successful misuse defense bars the misuser from prevailing against anyone on an action for infringement of the misused intellectual property, even against defendants who have not been harmed or affected by the misuse.[76]

The misuse doctrine was judicially created, first in the patent context. Only recently has the misuse doctrine been extended to copyrights, building on the rich misuse history in the patent law.[77] Importantly, most courts have found misuse without requiring a finding of antitrust liability.[78] Thus, market power is unnecessary, as is any analysis of the competitive and anticompetitive impacts of the provision.[79]

The courts have yet to analyze a copyleft provision for misuse, but the courts have addressed an analogous provision--the grantback. A grantback provision requires that a licensee of intellectual property grant back to the licensor a license or ownership in creations made by the licensee. The typical grantback provision requires that the licensee give the licensor a nonexclusive license to any improvements or derivatives that the licensee creates based on the original licensed property. The idea is that the licensee would not have been able to make the improvement or derivative without permission of the licensor or at least access to the original; thus, the licensor should not be blocked by an improvement or derivative he and his intellectual property helped create. Giving the license back ***369** encourages licensors to license, since it mitigates the risk of becoming blocked by derivative intellectual property. Like a grantback, copyleft requires the licensee to license back its improvements. The copyleft provision is more expansive, though. The grantback clause requires the licensee to give a license back to the licensor, while the copyleft license requires the licensee to give a license to the world.

Although grantbacks have not come up in the copyright misuse arena, they have in the patent context--and as we have seen, the patent misuse cases form the underpinning for the copyright misuse doctrine. Courts have found that grantback clauses extending to improvements are not misuse, because the licensee in some sense developed the improvement with the help of the original patent. Where grantback clauses extend to preexisting or unrelated patents, however, courts have found patent misuse. Where "the scope of [licensee's] 'improvements' and inventions required to be assigned to [the patent licensor] extended far beyond the scope of [the] basic patent [licensed by licensor] the effect was to extend unlawfully its monopoly and thus result in patent misuse."[80] Plainly, the Patent Act does not give the patent owner rights to other unrelated patents, and using a patent to obtain such rights exceeds the scope of the patent.

Similarly, the Copyright Act's grant of rights does not extend to unrelated works or preexisting (and therefore necessarily nonderivative) works, and using the copyright license to extract such rights exceeds the scope of the copyright grant. This may constitute copyright misuse. A license to a copyrighted work on condition that any work with which it is combined or shares data must be licensed back to the licensor--and the entire world--on the specific terms the licensor mandates, is beyond the scope of the copyright in the originally licensed work. Yet this is what the GPL apparently requires. The copyleft provision purports to infect independent, separate works that are not derivative of the open source code, and requires that such independent works be licensed back to the licensor and the entire world under the GPL. The Copyright Act does not give the copyright owner rights to such independent nonderivative works. Attempting to extract such rights exceeds the scope of the copyright. The fact that the GPL mandates that the license be free and ***370** open is irrelevant; as explained above, misuse doctrine does not require an analysis of market share, or a weighing of the competitive and anticompetitive effects of the provision.

If the copyleft provision constitutes misuse, then the plaintiff's copyrights in the open source program are unenforceable until the misuse is purged.[81] As a result, at least with respect to the code contributed by any plaintiff, the defendant (and anyone else) could infringe the copyright with impunity, including taking the code private for his own commercial ends.[82] Thus, licensors using copyleft licenses need to realize that they may be unable to enforce the copyleft provision against separate works of the licensee, and that any such attempt may at least temporarily invalidate all their copyrights in the entire open

source program. Copyleft licenses are still valuable, however, where they do not try to infect independent code. They should safely cover any dependent derivative works based on the original GPL code. Licensors simply need to understand the potential limitations and risks of copyleft to employ it effectively.

Yet even if fully enforceable, copyleft may still fail to prevent certain modifications from being taken private. The GPL permits modification of the code subject to the condition that the modification is distributed under the GPL. Similarly, the BSD license permits modifications provided certain conditions are met, including passing on notice of the license terms. In both cases, the license to modify is expressly conditioned upon making sure that the downstream licensees are bound by the same terms or notices. What if the user breaches the contract and fails to distribute under the open source license terms or notices? If these are truly conditions to the license to modify, then the right to create modifications is lost or never arose due to the failure of the condition (a condition subsequent, in this case). "You may modify your copy . . . provided that you also meet all of these conditions: . . . You must cause any work that you distribute . . . to be licensed . . . under the terms of this License."[83] In other words, for any work you distribute, the right to modify is only valid if you distribute the work under the GPL.[84]

***371** Accordingly, if you distribute the modifications under a license other than the GPL (or under no license), the modifications are thus not permitted, and are infringing derivatives. You would be a copyright infringer. The Copyright Act provides that the infringer does not own the infringing derivatives, but leaves open the question what happens to them--are they effectively owned by the underlying copyright owner whose rights are infringed, as a windfall bonus to any damages remedy (since the underlying copyright owner could claim that anyone else using these derivatives similarly infringed the underlying copyrights)? Or, do they instead fall into the public domain, free of the claims of the underlying copyright owner?[85] If the latter, then those infringing modifications that were supposed to be under the open source license are free to be taken private. Although the infringer may be enjoined from exploiting the infringing derivatives, as public domain works they are freely available to anyone else.

III. What to do about Open Source Licensing?

Open source licensing has become an important factor in the technology industry. Several responses to this development are available to commercial technology companies. One approach is to resist open source licensing, and try to stamp it out. Another approach is to work with the open source community. With an understanding of the implications of open source licensing, and an appreciation for the legal risks concerning enforceability of the license and the copyleft provision, open source licensing can be a valuable tool.

Microsoft's response illustrates the first approach. Microsoft Chairman Bill Gates, Senior Vice-President Craig Mundie, and President Steve Ballmer have waged a vocal campaign against open source software, branding it "a threat to our very system of capitalism."[86] Richard Stallman warns that Microsoft has explicitly targeted the open source community.[87] Indeed, Microsoft has hinted that it will deploy its intellectual property arsenal against the open source community. "The effect of patents and copyrights in combatting Linux remains to be investigated[,]" a Microsoft analyst wrote.[88] The open source community takes these threats ***372** seriously, particularly the threat from software patents. Richard Stallman acknowledges that "The worst threat we face comes from software patents, which can put algorithms and features off limits to free software for up to twenty years."[89]

Microsoft's motivation is either to stifle Linux, or co-opt it, according to the open source community. Linux presents a potential threat to the Windows operating system monopoly, and Microsoft would obviously like to see it disappear.[90] Alternatively, Microsoft may employ "an anticompetitive strategy called 'embrace and extend.' This means they start with the technology that others are using, add a minor wrinkle which is secret so that nobody else can imitate it, then use that secret wrinkle to make it so that only Microsoft software can communicate with other Microsoft software. . . . Hence their campaign to persuade us to abandon the license that protects our community, the license that won't let them say 'What's yours is mine, and what's mine is mine.'"[91] In other words, the copyleft provision of the GPL keeps Microsoft from co-opting Linux, taking it private, and then using its market share to drive adoption of its Linux as the standard. It would only be available from Microsoft, for a fee.

Thus, in addition to the attack on the open source community generally, Microsoft employs a specific attack on the copyleft provision. Microsoft has distributed software under a license that explicitly prohibits licensees from using the software with copyleft software.[92] Microsoft's license calls such software "Potentially Viral Software"--defined as any software which is

licensed under terms that create obligations for Microsoft or grant third parties rights in the Microsoft software.[93] Within this category the license specifically includes GPL, LGPL, Sun Industry Standards Source License (SISSL), as well as any software ***373** that is distributed as free software.[94] Microsoft fears that if GPL code is combined with or incorporated into Microsoft proprietary code, this would make Microsoft code also subject to the open license and the person creating the derivative would then be required to open source the Microsoft code with the incorporated GPL code. In fact, however, because a licensee cannot grant greater rights than he received from the licensor, a Microsoft licensee could not by combining the Microsoft code with GPL code lawfully release the Microsoft code under the GPL. Were it otherwise, anyone could simply incorporate some GPL code in Windows, and then claim Windows had become GPL code against Microsoft's wishes. Thus, Microsoft's anti-copyleft license restriction is not necessary, but it nevertheless precludes the licensee from using Microsoft software or distributing it "in conjunction" with any open source or free software.[95] Whether this is an attempt to hold back the open source community, or whether Microsoft ultimately succeeds, its strategy is only useful to those software companies with significant market share in the segment they wish to control.

For most companies, a better option is to work with the open source community and to co-opt the open source model as a facet of an overall licensing and business strategy.[96] One model, advocated by OSI, is to create a services business and give the open source software away for free. This is RedHat Software's business model. RedHat makes binary copies of the open source Linux operating system, and sells them in individually packaged boxes bearing the RedHat trademark. You can pick up a box of RedHat Linux at your retail software outlet. Although the Linux source code is freely available, RedHat is able to charge a modest amount for the box of software. RedHat saves users the cost and effort of compiling the source code into object code form, and perhaps most importantly, by obtaining the software from RedHat instead of one of the community websites, the user knows there is a specific entity to go to for support or questions or problems. Branding is therefore critically important to make a business model of selling open source software, because anyone can get it for free from other sources. Indeed, you or I could get the source code for RedHat's Linux (the GPL requires that RedHat ***374** make it available), compile it ourselves and sell it in competition with RedHat, as long as we refrain from violating RedHat's trademark.[97]

Thus, RedHat's revenue model is really centered on selling services to its users--support and maintenance, for example.[98] Since RedHat needs to spend very little to create the code (thanks to the efforts of the Linux developer community), the services business does not need to offset any material research and development costs.

OSI proposes a number of business models for taking advantage of open source software: first, give away the software and charge for services such as distribution of copies, support of the software, or performing related consulting work (the RedHat model); second, give away the open-source software as a loss leader to help your related proprietary software product gain mindshare, and leverage the open source community's development of new complementary open source software; third, sell hardware products and give away the open source software that runs on that hardware--this also encourages the open source community to further develop and improve that software (and hence the performance of your hardware), as well as create complementary products.[99] Each of these tactics can be employed by a proprietary technology company as a strategy to ultimately benefit its commercial business.

Open source licensing is also useful to establish an industry standard. Giving the software away for free under an open source license enables the open source community to work on and enhance the code. Moreover, since the software is not in control of any one party, and because it is free, a good program can quickly and safely become an industry standard. This is particularly effective when offering proprietary software that depends upon that standard, because establishing that standard will enable more sales of the proprietary software. Alternatively, this approach can help dissipate the market power of a competitor whose proprietary standard has become the industry standard. A proprietary industry standard enables that competitor to "rig" the standard to favor his products. Open source licensing is a weapon to undermine that advantage.

In addition to these business models based on the existing open source licenses, a clever developer can use open source licensing in other ways, or modify the license itself. One such approach is a dual track distribution model. One track is to release your original code under the GPL, take advantage of the free talent in the open source community and the robustness of peer-reviewed software, and gain ***375** market share from the usual proliferation of products based on the open source software. The second track is to release that same code simultaneously as a proprietary, official object code release of the software.[100] The proprietary release may also include some of your own additional enhancements and improvements which are not shared under the open source release. The key to this model is requiring that, for any modifications contributed by the community for inclusion in the official code base, those contributors must assign their copyright in the modification to you, the original developer. As the copyright owner, you can release the code simultaneously under the GPL and a proprietary

license. Since by definition here there are no upstream copyright owners for any part of the code (you are the original developer), you are not bound by the GPL except to the extent you choose to bind yourself--in other words, for the one copy of the code you distribute under the GPL.[101]

This model makes it easier to maintain a single official object code release of the program, since no one else can ensure that his competing release matches the official release. This is particularly important for software programs whose value depends on smooth interoperability with other copies of that program or other software. This model is also helpful for safely bundling additional software with the proprietary copy of the open source program, without any fear of infecting the additional software with the GPL. In addition, some customers may dislike open source software, and want to obtain only proprietary software (either for their internal use, or for incorporation into one of the proprietary products they sell). Having a dual track model means that there is a proprietary version of the program for such customers, which can be priced much more aggressively than the open source version, since it allows them to maintain a proprietary model for their own software products.[102]

Another approach to the standards problem is illustrated by the Sun Industry Standards Source License (SISSL), which is an approved OSI open source license.[103] The license is designed to encourage the industry to follow a specific ***376** standard or specification, although it does not absolutely require such conformance. Like the GPL or BSD license, it permits free modification and use of the SISSL code.[104] Like GPL, the original SISSL code may be distributed only under the SISSL.[105] This is a copyleft-like provision, to prevent the original code from being taken private. Unlike GPL, however, modified versions of SISSL code may be distributed "under a license of Your choice" provided that the modified program complies with the compatibility requirements for that SISSL code.[106] If the modified version does not comply with these requirements and thus does not conform to the standard, then the SISSL requires the licensee to publish the source code for those modifications under SISSL terms, or publish a reference implementation of those modifications together with the deviations from the standard, under SISSL terms.[107]

Thus, the SISSL's copyleft terms apply only to the original code, and to any modified versions that fail to conform to the standard. Modified versions that do conform to the standard may be distributed under a proprietary model. This allows companies to compete on the merits of their implementations of the standard, without fragmenting the standard itself. Unlike GPL code, these companies can differentiate their products by keeping their enhancements private; they are not required to share them as long as they meet the standard. Although the community could collectively decide to change the standard, as long as the change is implemented by the community, it should be open and effective (a standard is only useful if it is uniformly followed). The SISSL prevents anyone from taking the standard itself private, no matter how much market share they might have in their proprietary implementation. Finally, safely avoiding the copyright misuse problem of the GPL, the SISSL clearly limits itself to covering derivatives, and does not infect separate programs: "You may [combine] Original Code with other code not governed by the terms of this License and distribute [the combined work] as a single product. In such a case, you must make sure the requirements of this License are fulfilled for the Original Code."[108] This means that the separate work can still be distributed under a proprietary license, as long as the open source code with which it is combined remains under the open source license.

Open source licensing does not usually succeed as a product-oriented revenue model, however. With a copyleft license, you must share any improvements to the open source code. Thus, it becomes impossible to differentiate your products from your competitors' products, because you have to share with them any of those potentially differentiating improvements. In theory, the developer could gain a first mover advantage, since the developer could share the modifications at the same ***377** time it shipped the binary products for sale on the market. Yet it would not take long for competitors to incorporate those improvements and release identical products. Open source licensing is best used in conjunction with a commercial business model; it does not simply substitute for a proprietary business model.[109]

IV. Conclusion

This Article describes a number of different business models based on different uses of open source licensing. A skilled practitioner can help her software company clients work safely with open source code and use open source licensing to accomplish their business objectives. By avoiding any undesired open source tainting of one's proprietary software, addressing the potential holes in the current open source licenses, and accounting for the limitations of the copyleft provision, open source licensing can be an effective tool.

The Open Source Initiative supports such use of open source licensing in the business world.[110] Richard Stallman disagrees: "I would like to see proprietary software disappear because people think the idea is so disgusting, they wouldn't have anything to do with it."[111] Stallman's Free Software Foundation opposes commercial use of open source licensing on ethical grounds. Although the open source community is divided on this point, commercial enterprises should not be. Open source licensing has become an important part of the technology industry, and to attack it or resist it is to ignore an opportunity to use it for commercial advantage. Only the most myopic of companies, or the most dominant, would choose to forego such an opportunity.

Footnotes

a1 Director and Associate General Counsel, Sun Microsystems, Inc., and Adjunct Professor, University of California Berkeley Boalt Hall School of Law. This Article represents the views and analysis of the author alone, and not of Sun Microsystems, Inc. The author would like to thank Bill Lard, Mark Lemley, Sean Hogle and Michele Milnes Nadan for their generous assistance and thoughtful comments and suggestions.

1 Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1243, 219 U.S.P.Q. (BNA) 113, 115 (3d Cir. 1983).

2 Id., 219 U.S.P.Q. at 115-16. See also Data General Corp. v. Grumman Systems Support Corp., 825 F.Supp. 340, 354 n.7, 28 U.S.P.Q.2d (BNA) 1481, 1491 n.7 (D.Mass. 1993), aff'd 36 F.3d 1147, 32 U.S.P.Q.2d (BNA) 1385 (1st Cir. 1994).

3 Apple, 714 F.2d at 1243, 219 U.S.P.Q. at 116.

4 Id. Object code is also called "binary code."

5 Obviously, if there is a problem arising from the part of the original program that the end user changed on his own, it will be hard for the original developer to figure out what went wrong and fix it.

6 Data General, 825 F.Supp. at 359, 28 U.S.P.Q.2d at 1496 (distribution in object code form preserves trade secret status of the program's source code); Cybertek Computer Prods., Inc. v. Whitfield, 203 U.S.P.Q. (BNA) 1020, 1022 (Cal. Super. Ct. 1977) ("While there may have been a question in the past as to whether or not computer software is susceptible of protection under the trade secret doctrine, it is clear that such protection is available today in practically all jurisdictions.").

7 Apple, 714 F.2d at 1243, 219 U.S.P.Q. at 116; Mark A. Lemley & David W. O'Brien, Encouraging Software Reuse, 49 Stan. L. Rev. 255, 269 n.91 (1997). Some companies share source code with particular customers, under strict nondisclosure and limited use agreements.

8 Anthony J. Natoli, Open Source Software, 5 Multimedia & Web Strategist 2 (June 1999).

9 U.S. v. Microsoft Corp., 84 F.Supp.2d 9, 23 (D.D.C. 1999).

10 Open Source Initiative, OpenSource.Org, at http:// www.opensource.org/index.html (visited Aug. 17, 2001) [hereinafter OpenSource.Org]. For a discussion of the benefits of making code available for reuse, see Lemley & O'Brien, supra note 7, at 259-68.

11 Charles H. Cella & Edward J. Kelly, Considerations for Companies Developing Software Under the Open Source Model, 4 Cyber. Law. 9 No. 6 (1999).

12 Id.

[13] Richard Stallman, The GNU GPL and the American Way, at http:// www.fsf.org/philosophy/gpl-american-way.html (visited Aug. 17, 2001) [hereinafter The GNU GPL and the American Way]. The Free Software Foundation, a tax-exempt charity, was founded in 1985. Richard Stallman, The GNU Project, at http://www.fsf.org/gnu/thegnuproject.html (last modified Dec. 2001) [hereinafter The GNU Project].

[14] Id. ("Computer users should be free to modify programs to fit their needs, and free to share software, because helping other people is the basis of society.").

[15] More colorfully, Stallman advises "you should think of 'free' as in 'free speech,' not as in 'free beer.'" Free Software Foundation, The Free Software Definition, at http://www.fsf.org/philosophy/free-sw.html (visited Aug. 17, 2001) [hereinafter The Free Software Definition].

[16] The GNU GPL and the American Way, supra note 13. Stallman's personal experience shaped his views: "The laser printers of the mid-1970's were the size of today's compact cars. When Xerox gave the AI lab a Xerox Graphics Printer, the only place for it was in the lab's ninth-floor machine room. Researchers connected the printer to the local area network that the lab was developing, and soon anybody in the building could print a 100-page document by typing in a few commands. That worked fine, except that sometimes the printer would run out of paper or jam, and dozens of other jobs would pile up. Other times there would simply be a lot of people wanting to print long documents, and the person who needed to print a single page would have to run up and down the stairs or babysit the printer until that page appeared. But since the programmers at the lab had the source code to the program that ran the printer, they could add features that solved these problems. Soon the printer was helping the lab run smoothly. 'It would send you a message when your document had been printed,' recalls Stallman. 'It would send you a message if you had anything queued and there was a paper jam.' All this changed in 1978, when Xerox replaced the machine with a new laser printer called a 'Dover' but wouldn't share the printer's source code with the lab. 'We wanted to put those features into the Dover program, but we couldn't,' Stallman says. Xerox wouldn't put the features into the program either. 'So we had to suffer with paper jams that nobody knew about.'" Simson L. Garfinkel, Programs to the People, Tech. Rev., Feb./Mar. 1991, at 53, 54 (cited in Ira V. Heffan, Copyleft: Licensing Collaborative Works in the Digital Age, 49 Stan. L. Rev. 1487, 1504 n.101 (1997)).

[17] The GNU Project "was launched in 1984 to develop a complete UNIX-like operating system... GNU is a recursive acronym for 'GNU's Not Unix'; it is pronounced 'guh-NEW.'"). Free Software Foundation, GNU's Not UNIX!, at http:// www.fsf.org/ (visited Aug. 17, 2001).

[18] The GNU Project, supra note 13.

[19] See Marcus Maher, Open Source Software: The Success of an Alternative Intellectual Property Incentive Paradigm, 10 Fordham Intell. Prop. Media & Ent. L.J. 619, 621-24 (2000) (reviewing a few open source software programs in use today).

[20] Open Source Initiative, History of the OSI, at http:// www.opensource.org/docs/history.html (visited Aug. 17, 2001) [hereinafter History of the OSI].

[21] Id.

[22] Id.

[23] Id. See also OpenSource.Org, supra note 10 ("Open Source Initiative exists to make this case to the commercial world.").

[24] Open Source Initiative, Why Free Software is Too Ambiguous, at http://www.opensource.org/advocacy/free-notfree.html (visited Aug. 7, 2001) (Indeed, their greatest fear was a proprietary software company opposed to open source like "Microsoft claiming Internet Explorer is 'free software' because its cost is zero dollars. Would we really want that?").

[25] OpenSource.Org, supra note 10 ("Open Source Initiative (OSI) is a non-profit corporation dedicated to managing and promoting" open source licensing).

[26] Open Source Initiative, Advocacy, the Open Source Case for Business, at http://www.opensource.org/advocacy/case_for_business.html (visited Aug. 7, 2001) [hereinafter Advocacy]. Like OSI, the noted Internet commentator Lawrence Lessig agrees that the best societal approach is a combination of open and proprietary code. Lawrence Lessig, The Limits in Open Code: Regulatory Standards and the Future of the Net, 14 Berkeley Tech. L. J. 759, 767-68 (1999) (discussing how the functioning of the Internet is really determined by the code on which it is built, so the more that the Internet is built on open source code, the harder for governments to regulate or control it). See also Maureen O'Rourke, Symposium on Bioinformation and Intellectual Property Law, 8 B.U.J. SCI & TECH.L. 254 (Winter 2002) (discussing whether the open source model that promotes sharing in the software world can be applied to the biotechnology industry, to promote sharing of information such as in the human genome project).

[27] Id.

[28] Id.

[29] Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt (visited Aug. 7, 2001); University of California, Berkeley, Systems Research Group, The BSD License (July 22, 1999), at http://www.opensource.org/licenses/bsd-license.html (visited Aug. 7, 2001).

[30] Open Source Initiative, The Open Source Definition, at http:// www.opensource.org/docs/definition.html (visited Aug. 7, 2001) [hereinafter The Open Source Definition].

[31] Id.

[32] Open Source Initiative, The Open Source Initiative, The Approved Licenses, at http://www.opensource.org/licenses/index.html (visited Aug. 7, 2001) [hereinafter The Approved Licenses].

[33] The Free Software Definition, supra note 15.

[34] Open source development is therefore not a true commons, because open source licenses are used to enforce the community norms. David McGowan, Legal Implications of Open-Source Software, 2001 U. Ill. L. Rev. 241, 244-45 (discussing the sustainability of the open source model in light of the legal and organizational issues presented). The property rights protected by open source licensing are used not to exclude, however, but to ensure that no one else excludes anyone. As Professor McGowan explains: "like the sword of Damocles, the point is not that it falls but that it hangs." Id. at 303.

[35] Free Software Foundation, Licenses, at http:// www.fsf.org/licenses/licenses.html (visited Aug. 7, 2001).

[36] Free Software Foundation, Various Licenses and Comments About Them (Feb. 2000), at http://www.fsf.org/licenses/license-list.html ("If you want a simple, permissive, non-copyleft free software license, the modified BSD license is a reasonable choice.") (visited Aug. 7, 2001) [hereinafter Various Licenses and Comments About Them]. Technically the current BSD License is in fact a modified version of the original BSD License. Id. (For convenience I will refer to the modified BSD License throughout as the BSD License).

[37] Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt (visited Aug. 7, 2001).

[38] Id. § 1.

[39] Id. § 1 para. 2.

[40] Id. § 2(b).

[41] Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt at § 2(c) para. 3 (visited Aug. 7, 2001).

[42] Id. § 2(c) para. 2. The GPL also enables you to bundle your proprietary program safely on the same CD as a GPL program. Id. § 2(c) para. 4.

[43] Id.

[44] Id. § 2(b).

[45] For instance, IBM invests heavily in Linux, but recognizes that "using GPL-licensed software does involve some careful planning, to avoid any situation where a company might be forced to give away its own work--something that can happen if the GPL software becomes part of another product." Dan Gillmor, Big Blue Places $1 Billion Bet on Linux, San Jose Mercury News Aug. 15, 2001 at 6C.

[46] The point of this discussion is not to scare the reader away from open source licensing, but rather to emphasize the potential dangers and highlight the need for an informed approach to using open source licensing in a commercial enterprise.

[47] Open Source Definition, supra note 30, at annotation to #9.

[48] Some argue that this OSI interpretation is too extreme, and is inconsistent with acknowledged practice in the open source community. See Silicon Valley.com, SV.com Roundtables, Code War: The hot debate over free, open-, and shared-source software (a moderated online forum on open source licensing) [hereinafter Roundtable] at http://forums.siliconvalley.com/discussion/msgshow.cfm/msgboard=5968009897410465&msg=1431390223675614&page=1&idDispSub=5145094516046185 (visited Aug. 30, 2001). For a commercial enterprise, however, whether the OSI interpretation is too extreme or not is beside the point. The debate illustrates the obvious ambiguity in the text of the GPL, and makes such linking risky, at best. A commercial enterprise cannot afford to lose the test case on this point. See also The GNU Project, supra note 13. Stallman discusses "combining a free program with non-free code. Such a combination would inevitably be non-free; whichever freedoms are lacking for the non-free part would be lacking for the whole as well. To permit such combinations would open a hole big enough to sink a ship. Therefore, a crucial requirement for copyleft is to plug this hole: anything added to or combined with a copylefted program must be such that the larger combined version is also free and copylefted." Id.

[49] Of course, if you can find, isolate and remove the GPL code, for any cleaned copies you may be able to undo the requirement to license the proprietary product under the GPL. Bruce Perens, Roundtable, supra note 48, at July, 25, 2001, 4:56 a.m. posting ("GPL violations are remedied by removing the GPL code from the product.") (Perens is the author of the OSI Open Source Definition. See History of the OSI, supra note 20). On the other hand, the copies already shipped with GPL code cannot be saved. And it takes only a single copy of your code under GPL to perpetuate itself across the globe. This viral effect has also been likened to "tar: once you use it, it tends to stick to you." Lisa Green & Heather Meeker, Open Software Licenses, Part I, 5 Intell. Prop. Strategist 2 (June 1999). Linus Torvalds, the creator of the Linux kernel, offers an ambiguous comment on copyleft--"When the question was raised whether proprietary code could be tied to GNU/Linux... Torvalds said 'My opinion on licenses is that "he who writes the code gets to choose his license, and nobody else gets to complain. Anyone complaining about a copyright license is a whiner."" ' Lawrence Lessig, Open Code and Open Societies: Values of Internet Governance, 74 Chi.-Kent L. Rev. 1405, 1419-20 (1999).

[50] Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt at § 3 (visited Aug. 7, 2001).

[51] Id. §§ 11, 12. The strong viral effect of GPL is highlighted when contrasted with the less strong viral effect of the GNU Lesser General Public License (LGPL). Free Software Foundation, GNU Lesser General Public License, Version 2.1 (Feb. 1999), at http://www.fsf.org/licenses/lgpl.txt [hereinafter LGPL]. LGPL is similar to GPL, but is "not a strong copyleft license, because it permits linking with non-free modules." Various Licenses and Comments About Them, supra note 36. It is called the "Lesser" GPL because "it does Less to protect the user's freedom than the ordinary General Public License." LGPL at Preamble. Specifically, LGPL does not completely infect independently developed programs that work with the LGPL code by being "compiled or linked with it." Id. §§ 5, 6 (emphasis added). Such a combined work may be distributed "under terms of your choice," provided however that such terms "permit modification of the work for the customer's own use and reverse engineering [converting from object code to source code] for debugging such modifications." Id. § 6. Thus, the independent code does not become LGPL code, but it must be subject to modification by the licensee. This confirms the harsher effect of GPL described above--if the Lesser GPL partly infects independent code that is merely linked to open source code while the software is running (even though it is not actually compiled with the open source code into a single object code program), regular GPL must completely infect independent code that is merely linked to the GPL code while running. It becomes GPL code. Indeed, OSI interprets GPL just this way. The LGPL is thus encouraged only for libraries (a library is a "collection of software functions" which can be used by ("linked with") application programs), and only in a few circumstances (LGPL used to be called the Library GPL): (i) for the open source library to become a de facto standard, non-free code must be allowed to use it too; (ii) where a free library does the same job as a popular non-free library, there is little advantage to limiting use of the free library to free software only; (iii) widespread free and non-free programs' use of a free library may be needed if another free program relies on the ubiquity of such library. Id. at Preamble § 0.

[52] The BSD License, supra note 29. It also includes a disclaimer of warranties and a provision absolving the developer of the program and all contributors from any liability. Id. This means that in addition to the copyleft and viral risks, there are other more traditional risks in using open source software: it is typically provided with no warranties, no support, unclear title (since there are many contributors of varied integrity and sophistication), and no indemnity if the code infringes a third party's intellectual property rights.

[53] A "shrinkwrap" license is a license agreement that is usually contained in a box of software, which states that by opening the package, you agree to the terms of the license agreement. By way of comparison, a "clickwrap" agreement is often found on an internet website, and the user is required to click an "accept" button at the end of the license agreement before the software can be downloaded (some people call this a "webwrap"). A clickwrap can also be a license that first pops up on the screen when the software is actually being installed on the user's computer, requiring the user to click an "accept" button before the installation will conclude (sometimes called an "installwrap" license); if you decline to accept, the installation will abort.

[54] Klocek v. Gateway, Inc., 104 F.Supp.2d 1332 (D.Kan. 2000). Note that this case refusing to enforce a shrinkwrap license involved passive conduct to indicate acceptance, as in GPL or BSD, instead of requiring affirmative acts to indicate acceptance like the tearing open of a shrinkwrapped package, or clicking "accept" to a clickwrap presented on installation or booting up of the software. Using passive conduct (like failing to return the computer within 30 days of purchase) to indicate acceptance creates uncertainty whether the licensee even knew his conduct was creating a binding obligation, and casts doubt on the enforceability of the purported agreement. See also Step-Saver Data Sys., Inc. v. Wyse Techn., 939 F.2d 91 (3d Cir. 1991) (refusing to enforce shrinkwrap agreement).

[55] The BSD License, supra note 29.

[56] The GPL does say that 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." Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt § 5 (visited Aug. 7, 2001). The problem is that the user is never forced to see the GPL, and so may not see this term. The issue here is not whether the licensee fails to read the license, but whether he knows of its existence at all.

[57] Ironically, the GPL does require that the copyright notice and disclaimer of warranty be "conspicuous," but not the notices of the GPL or the GPL itself. Id. § 1 para. 1. Note that Section 2b of the GPL provides that any modified GPL code must be distributed in accordance with the terms for unmodified code (set out in Section 1), and "provided that you also meet all of these conditions," one

of which is that it must be licensed under the terms of the GPL. If this is held to be a condition to the license grant, then any distribution of modified code not under GPL is a failure of this condition, and so the conditional license never becomes effective. This would mean that the distribution is in violation of the license, and the sublicensee has no more rights than did the licensee--in other words, the sublicensee has gained possession of an infringing copy, and has no right to use or distribute it. For some reason, the GPL terms for distributing unmodified code are not subject to this additional condition that the distribution must be licensed under GPL. It should be noted, however, that conditions are disfavored, and if the purported condition is determined to be only a covenant, then even deliberate release of GPL code under a non-GPL license would be effective--the distributor would be subject to a simple breach of contract claim, but his sublicensee would not be subject to any GPL terms, and free to use the formerly GPL code however it was licensed to him by the distributor. Id. § 2. See Graham v. James, 144 F.3d 229, 237, 46 U.S.P.Q.2d (BNA) 1760, 1765-66 (2d Cir. 1998).

58 54 U.S.P.Q.2d (BNA) 1344 (C.D. Cal. 2000).

59 Id. at 1346.

60 Specht v. Netscape Comm. Corp., 150 F.Supp.2d 585, 595 (S.D.N.Y. 2001).

61 Compare Storm Impact, Inc. v. Software of the Month Club, 13 F.Supp.2d 782, 48 U.S.P.Q.2d (BNA) 1266 (N.D. Ill. 1998), where the court found that a legend which appears on the screen display of a computer program imposed enforceable contractual restrictions (prohibiting commercial redistribution) even though no one at the defendant company "either read or paid attention to the express restrictions." Id. at 786, 48 U.S.P.Q.2d at 1268. Like open source software, shareware is software distributed for free, although usually with a request that some money be sent to the developer upon actual use of the software. Id. at 785, 48 U.S.P.Q.2d at 1267. But unlike Ticketmaster, here there was no claim that defendants had not seen the notices, and defendant conceded they were binding in any event by admitting that violating the restrictions constituted copyright infringement. Id. at 787, 48 U.S.P.Q.2d at 1269. A recent decision confirms that Terms of Use are enforceable where the user does have notice. Register.com, Inc. v. Verio, Inc., 126 F.Supp.2d 238 (S.D.N.Y. 2000). Register.com is an authorized registrar of internet domain names, and its website provides query-based access to the identities of registered domain name owners. (You enter a particular domain name, and the site will tell you who registered that domain name.) Register.com "clearly posted" on its website Terms of Use stating that by submitting a query for information, "you agree to abide by these terms." Id. at 248. The Terms prohibited use of the information for mass marketing purposes. Verio nonetheless queried the website (using a search robot) for information which it then used for mass marketing purposes. Verio did not argue that it was unaware of the Terms of Use when it submitted its queries to the plaintiff's website, and that was enough to create a contract, the court said. Id. It was not necessary for Verio to have taken any further steps to manifest assent, such as clicking an "I Agree" button. Verio was aware of the Terms of Use--which said that submitting a query bound you to such terms--so that when Verio submitted a query it manifested assent to the Terms of Use. Id.

62 Specht, 150 F.Supp.2d at 595.

63 Id. See Softman Products Co., LLC v. Adobe Systems, Inc., 171 F.Supp.2d 1075, 1087 (C.D. Cal. 2001) (software license appeared only during installation of the software product--and since Softman never installed the software, it never saw the license and is thus not bound; moreover, the existence of a notice on the software box that "This product is offered subject to the license agreement included with the media" did not make any difference).

64 Indeed, for an agreement to be binding under UCITA (the Uniform Computer Information Transactions Act), by way of example, the customer must have actual knowledge of the proposed contract, or the contract must have been made available to the customer in a manner that "ought to call it to the attention of a reasonable person and permit review." UCITA §§ 112(a), (e) (1999 Official Text). UCITA has been vilified by consumer groups and the FTC, as far too pro-business. Interestingly, as much as the free source movement wants to distance itself from pro-business interests, these two groups are effectively aligned on UCITA--both groups want to codify shrinkwrap licensing, make warranties easy to disclaim, and retain the model that software is licensed, not sold. See Robert W. Gomulkiewicz, How Copyleft Uses License Rights to Succeed In the Open Source Software Revolution and the Implications For Article 2B, 36 Hous. L. Rev. 179 (1999) (arguing that the open source movement needs proposed UCC Article 2B (which became UCITA)). The Free Software Foundation does claim to oppose UCITA, however. The Free Software Foundation, Why We Must Fight UCITA, at http:// www.fsf.org/philosophy/ucita.html (visited Aug. 7, 2001).

65 In addition, making open source code available for download over the internet is essentially making it available all over the world. It may be downloaded by users in any country, including some whose local laws may not enforce shrinkwrap licenses, even if the downloader does have notice of the license. Jorge Contreras, Jr. & Kenneth H. Slade, Click-Wrap Agreements: Background and Guidelines for Enforceability, Computer und Recht Int'l 104, 108 (April 2000) (reviewing global enforceability of shrinkwrap and clickwrap licenses). The global enforceability of shrinkwraps is beyond the scope of this Article, but it is clear that the GPL's notice-in-the-code form of licensing may fail to bind the downloader, and fail to preclude his taking the code private.

66 Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt § 5 (visited Aug. 7, 2001).

67 Jay Dratler, Jr., Licensing of Intellectual Property § 3.04[1] (9th ed. 2000) (citations omitted).

68 Id.; Graham, 144 F.3d at 235, 46 U.S.P.Q.2d at 1764 ("'non-exclusive licenses may... be granted orally, or may even be implied from conduct.'") (citing 3 Melville B. Nimmer & David Nimmer, Nimmer on Copyright § 10.03[A][7], at 10-43 (2001)).

69 See Dratler, supra note 67, §§ 3.04- 3.05; Specht, 150 F.Supp.2d at 595 (although clickwrap license agreement did not bind the downloaders, no one contended that the downloaders' continued use of the software was without authorization or that they lacked an implied license; the court "analogized to a free neighborhood newspaper, readily obtained from a sidewalk box or supermarket counter without any exchange with a seller or vender. It is there for the taking.").

70 Since the licenses are free for everyone, the lack of payment to the licensor should not preclude the creation of an implied license. In addition, the disclaimer of warranties and the limitation of liability in the open source license would not apply. Although the licensor would not be shielded from any damages liability, the implied warranties of merchantability and fitness for a particular purpose will not apply unless the licensor is a merchant of that software, or knows of the licensee's particular purpose for the code and is relying on the licensor's skill in selecting that code, respectively. U.C.C. §§ 2-314, 2-315 (2001). It is less certain that the implied license would include the right to redistribute in addition to the right to use the code. With source code made freely available for download without any restrictions (as far as the downloader is aware), however, a court might likely find an implied right to distribute as well. Professor Lemley explains that "Courts will regularly imply a license to exercise one or more of the exclusive rights granted the copyright owner in circumstances where to do otherwise would defeat the reasonable expectations of the implied licensee." Mark A. Lemley, Dealing With Overlapping Copyrights on the Internet, 22 U. Dayton L. Rev. 547, 580 (1997). Unlike object code, which is more typically provided as a finished product for end use, source code enables easy manipulation and incorporation into other software, and making software available in such form would suggest an authorization to use it for those purposes. In fact, if no license is apparent it would appear to be free to use in a distributed product as much as an internally used one. Where the licensor fails to make the downloader aware of the license, the licensee's likely expectation--that the source code made available on the internet can be used and distributed--is reasonable. Indeed, the licensor did make the code available for this express purpose--modification and redistribution. The licensor intended that downloaders be able to do exactly this; the only understanding the licensor and implied licensee did not share was whether such modification and distribution would be subject to a written but unseen open source license. The court would be following the parties' mutual intent in finding an implied right to modify and redistribute the code. Given that the licensor intended that the downloader be able to distribute the code, it would be unfair to permit the licensor to sue the downloader for doing just that, based on the licensor's own failure to notify the downloader of the license. See also supra note 69 and accompanying text; McGowan, supra note 34, at 298-99 (where no term is stated (the GPL does not specify any term), the license may be terminable at will. Thus, downloaders may "be better off if the licenses were unenforceable... for then the [downloaders] could rest on estoppel arguments" [like an implied license] to prevent such termination).

71 Microsoft Corp. v. Harmony Computers & Electronics, Inc., 846 F.Supp. 208, 214, 31 U.S.P.Q.2d (BNA) 1135, 1139 (E.D.N.Y. 1994) ("If defendants purchased their Products from Microsoft licensees who were acting outside the scope of their licenses... any distribution of the Products by defendants, whether within the scope of [Microsoft]'s license agreement or not, would constitute copyright infringement."); Adobe Systems Inc. v. One Stop Micro, Inc. 84 F.Supp.2d 1086, 1093, 53 U.S.P.Q.2d (BNA) 2003, 2008 (N.D.Cal. 2000) (noting that "by obtaining Adobe software from a party to an Adobe licensing agreement, One Stop was bound by any restrictions imposed by that agreement" even though One Stop had no contractual relationship of any kind with Adobe). Cf. Softman Products Co., LLC v. Adobe Systems, Inc., 171 F.Supp.2d. 1075, 1087-88 (C.D. Cal. 2001) (declining to adopt analysis of Harmony and One Stop).

72 Alternatively, the GPL could be modified so that distribution of the GPL code is subject to the condition that it is licensed under

the terms of the GPL. The GPL already includes this condition for distributions of modified code, which may protect such code from the unenforceability problem discussed here. See supra note 57 and accompanying text.

[73] Brulotte v. Thys Co., 379 U.S. 29, 32, 143 U.S.P.Q. (BNA) 264, 266 (1964) (holding that an obligation to pay royalties in return for use of the patented device may not extend beyond the life of the patent).

[74] 35 U.S.C. Section 154(a)(2) (1994).

[75] Morton Salt Co. v. G.S. Suppiger Co., 314 U.S. 488, 492, 52 U.S.P.Q. (BNA) 30, 32 (1942) (refusing to enforce a patent on a salt depositing machine where the patent owner required all users to purchase only his form of unpatented salt tablets to use in the machine).

[76] Lasercomb America, Inc. v. Reynolds, 911 F.2d 970, 979, 15 U.S.P.Q.2d 1846, 1854 (4th Cir. 1990) (noting that the Morton Salt defendants themselves were not parties to the license containing the offending clause). See also Practice Mgmt. Info. Corp. v. American Medical Assoc., 121 F.3d 516, 43 U.S.P.Q.2d (BNA) 1611 (9th Cir. 1997) (involving a free, terminable-at-will software license which required that a licensee use only the plaintiff's copyrighted set of medical codes. Although there was no evidence the licensee had any issue with the license, a third party defendant successfully asserted the misuse defense against a claim that it infringed the copyrighted codes. The court found that since the license exceeded the scope of the copyright, the copyright had been misused. No other factors were discussed).

[77] Lasercomb, 911 F.2d at 977, 15 U.S.P.Q.2d at 1852 (finding copyright misuse doctrine "analogous to the misuse of patent defense").

[78] Lasercomb, 911 F.2d at 978, 52 U.S.P.Q. at 1853 (stating that "it is not necessary to prove an antitrust violation in order to successfully assert patent misuse"). Practice Mgmt., 121 F.3d at 521, 43 U.S.P.Q.2d at 1615-16 (stating that one "need not prove an antitrust violation to prevail on a copyright misuse defense.").

[79] The one exception is with respect to a claim of misuse based on tying of a patent or patented product--the 1988 Patent Misuse Reform Act imposed a requirement that misuse cannot occur based on patent tying unless the plaintiff had market power in the relevant market for the patent or patented product. 35 U.S.C. Section 271(d)(5) (1994).

[80] Duplan Corp. v. Deering Milliken, Inc., 444 F.Supp. 648, 699-700, 197 U.S.P.Q. (BNA) 342, 388 (D.S.C. 1977) aff'd in part and rev'd in other part, 594 F.2d 979, 201 U.S.P.Q. (BNA) 641 (4th Cir. 1979); Transitron Electronic Corp. v. Hughes Aircraft Co., 487 F.Supp. 885, 893, 904, 205 U.S.P.Q. (BNA) 799, 806, 815 (D. Mass. 1980), aff'd 649 F.2d 871, 210 U.S.P.Q. (BNA) 161 (1st Cir. 1981) (including "overbroad grant-backs" as one "of the classic forms of patent misuse," and noting that "the Supreme Court held that a license provision requiring that the licensee 'grant back' all its own patents to the licensor might constitute patent misuse and bar the enforcement action."); Cf. Robintech, Inc. v. Chemidus Wavin, Ltd., 450 F.Supp. 817, 822, 197 U.S.P.Q. (BNA) 657, 660 (D.D.C. 1978), aff'd 628 F.2d 142, 205 U.S.P.Q. (BNA) 873 (D.C. Cir. 1980) (finding no misuse where grantback clause was not enforced and the contract language did not plainly extend the grantback obligations to licensee improvements that were unrelated to the original licensed patent). A finding of misuse is more likely with a grant back requiring an assignment back to the licensor (where the licensee has only a license to his own work), rather than a license back. Yet copyleft grantback licenses are more like an assignment back, in that copyleft denies the licensee the right to do what he pleases with his work; instead, he is allowed to license his own work only on terms dictated by the original licensor, and the licensor is free to license the licensee's work to whomever she wants, without any compensation to or consent from the licensee. From the perspective of the licensee, this is much more like having retained only a license than having retained ownership.

[81] One could argue that the misuse would not render the third party copyrights in the code unenforceable, but since every contributor to the GPL code used the same copyleft provision, any one of those contributors trying to enforce the copyright in the same court might face the same misuse result. Note that the Free Software Foundation encourages the public to report any violations of its copyleft licenses. Various Licenses and Comments About Them, supra note 36.

[82] One commentator discusses the misuse doctrine and open source licensing, but for a different purpose. Rather than suggesting the

misuse doctrine could enable the licensee to evade copyleft, he advocates using the misuse doctrine to bar the licensor from evading copyleft and taking its code private, once it becomes a standard. Chip Patterson, Copyright Misuse and Modified Copyleft: New Solutions to the Challenges of Internet Standardization, 98 Mich. L. Rev. 1351 (2000).

[83] Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http://www.fsf.org/licenses/gpl.txt § 2 (visited Aug. 7, 2001).

[84] See Sun Microsystems, Inc. v. Microsoft Corp. 188 F.3d 1115, 1121, 51 U.S.P.Q. (BNA) 1825, 1830 (9th Cir. 1999) (stating that if a condition to the license grant is not satisfied, then any licensed use of the software is infringing, and the license grant is no defense). This condition would help in the event of a licensee who deliberately passes on the software without an open source license. If the distribution was conditioned upon it being covered by an open source license, then it is an unlawful copy, and the copyright owner could probably sue the innocent downstream user for copyright infringement, even though any contractual privity would be lacking. But see Robert P. Merges, The End of Friction? Property Rights and Contract in the "Newtonian" World of On-Line Commerce, 12 Berkeley Tech. L. J. 115, 129 (1997) (arguing that the copyleft provision is unenforceable against downstream licensees in the first place, since they do not have contractual privity with the original contributor, but noting that the copyleft provision is nonetheless followed as a "non-binding, informal norm in cyberspace").

[85] 17 U.S.C. § 103(a) (1994). See Sean Hogle, Unauthorized Derivative Source Code, 18 The Computer & Internet Lawyer 1 (May 2001) (discussing ownership of unauthorized derivative works under copyright and patent law).

[86] Stephen Shankland, Microsoft License Spurns Open Source, CNET News.com, June 22, 2001, at http://news.cnet.com/news/0-1003-200-6352301.html? tag=tp_pr, (visited August 9, 2001); Gillmor, supra note 45, at 1C.

[87] Is Microsoft the Great Satan?, at http:// www.fsf.org/philosophy/microsoft.html (visited Aug. 17, 2001).

[88] Halloween Document II 1998, at http:// www.opensource.org/halloween/halloween2/html, quote 9 (visited Aug. 7, 2001). The so-called Halloween Documents are believed to have leaked from a Microsoft employee and incorporate the contributions of many senior Microsoft executives. See Halloween Document III at http:// www.opensource.org/halloween/halloween3.html (a Microsoft document conceding that the Halloween Documents appear to have been "written within Microsoft in August.") (visited Aug. 7, 2001); Maher, supra note 19, at 682 n.296 (Halloween Documents are from Microsoft).

[89] The GNU Project, supra note 13; The GPL Preamble notes that "any free program is threatened constantly by software patents." Free Software Foundation, GNU General Public License, Version 2 (June 1991), at http:// www.fsf.org/licenses/gpl.txt at Preamble para. 7 (visited Aug. 7, 2001). Patents are only available for new and nonobvious inventions, so the GPL model actually helps fight the threat of software patents. 35 U.S.C. §§ 101-03. By requiring open sharing of modifications, it establishes a greater volume of published prior art, used for proving that alleged patents are in fact not new or non-obvious. Nigel Howard, Dan Ravicher & Ken Johnson, How to Use Patent Law to the Advantage of Open Source Software Developers, 7 Intell. Prop. Strat. 1 (May 2001).

[90] "[S]ome observers believe Microsoft is worried that Linux--the best-known open-source project--will undermine the Microsoft.Net strategy." Shankland, supra note 86. Cf. U.S. v. Microsoft Corp., 84 F.Supp.2d at 23 ("It is unlikely, though, that a sufficient number of open-source developers will commit to developing and continually updating the large variety of applications than an operating system would need to attract in order to present a significant number of users with a viable alternative to Windows."). But see Dan Gillmor, Lindows Could Give Linux Life, And Worry Microsoft, SAN JOSE MERCURY NEWS, January 9, 2002, at 1C (discussing "Lindows," a new version of Linux designed to be able to run major applications that are written for Windows).

[91] The GNU GPL and the American Way, supra note 13.

[92] Shankland, supra note 86 (including link to the actual license clause).

[93] Id.

[94] Id.

[95] Id.; Perens observes that "[s]oftware developers, upon reading the Microsoft license, will wonder if they must now purge their establishments of all publicly available software tools so as not to run awry of Microsoft's licensing--and this is exactly what MS intends the license to compel them to do: get rid of [their open source software]." Roundtable, supra note 48, at July 25, 2001, 9:17 p.m. posting. Indeed, Microsoft appears to concede that the anti-copyleft restriction is overbroad or unnecessary, and promises to "revise the license terms, based on [the Roundtable's] feedback." Craig Mundie, Roundtable, supra note 48, at July 27, 2001, 5:39 p.m. posting.

[96] In fact, the three authors who have contributed the most open source code are institutions--the Free Software Foundation, Sun Microsystems, and the University of California. McGowan, supra note 34, at 276. In addition, IBM is reportedly investing $1 billion in Linux. Gillmor, supra note 45, at 1C. Even the U.S. government has released an open source software program. Developments in Brief, 43 No. 37 GOV'T CONTRACTOR paragraph 383(g) (October 10, 2001).

[97] "[A]lthough we retain the copyrights to the code we develop ourselves, due to the open source nature of our software products and the licenses under which we develop and distribute them, our most valuable intellectual property is our collection of trademarks." RedHat, Inc., Amendment No. 5 to Form S-1, 53 (Aug. 11, 1999) (cited and discussed in McGowan, supra note 34, at 252).

[98] Advocacy, supra note 26.

[99] Id.; Elise Ackerman, Penguin Power, The San Jose Mercury News, Nov. 25, 2001, at 1F, 8F (large technology companies do not need to make money selling Linux; they use it to help sell hardware running Linux and complementary proprietary software).

[100] This would enable the developer to avoid RedHat's problem--the inability to differentiate its product since everyone has access to the same code. Although "RedHat has access to improvements made by other persons and firms.... RedHat must give other persons and firms access to its own improvements." McGowan, supra note 34, at 252.

[101] LGPL validates a dual distribution model, although in the reverse direction. LGPL provides that the licensee "may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library.... Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy." LGPL, supra note 51, at section 3 (emphasis added). This proprietary-for-fee/open-for-free dual distribution model can also help establish a measure of damages in an infringement suit concerning a patent covering the software, since the fee model will establish a reasonable royalty rate. See also Howard, Ravicher & Johnson, supra note 89.

[102] Perens, Roundtable, Part 3 (noting as an example that the developer of the ReiserFS filesystem software retains ownership of the copyrights on the entire code base, and releases it under the GPL and also under a commercial proprietary license).

[103] The Open Source Initiative, The Sun Industry Standards Source License (SISSL), at http://opensource.org/licenses/sisslpl.html (visited March 30, 2002).

[104] Id. § 2.

[105] Id. § 3.1.

[106] Id. § 3.1, 3.3.

[107] Id. § 3.1.

[108] Id. § 3.4.

[109] Perens, Roundtable, supra note 48, at July 25, 2001, 9:17 p.m. posting ("Trying to do a straight software sales business using Open Source can be pretty tough--a lot of people have failed at that, and more will.").

[110] Advocacy, supra note 26.

[111] John Foley, The High Priest of Software, Informationweek, Aug. 14, 1995, at 36 (cited in Heffan, supra note 16, at 1505 n.102).