


2000

Opening Up to Open Source

Shawn W. Parker

Follow this and additional works at: <http://scholarship.richmond.edu/jolt>

 Part of the [Intellectual Property Law Commons](#), and the [Internet Law Commons](#)

Recommended Citation

Shawn W. Parker, *Opening Up to Open Source*, 6 Rich. J.L. & Tech 24 (2000).

Available at: <http://scholarship.richmond.edu/jolt/vol6/iss5/5>

This Article is brought to you for free and open access by UR Scholarship Repository. It has been accepted for inclusion in Richmond Journal of Law and Technology by an authorized administrator of UR Scholarship Repository. For more information, please contact scholarshiprepository@richmond.edu.

Volume VI, Issue 5, Spring 2000

Opening Up to Open Source

Shawn W. Potter^[*]

Cite As: Shawn W. Potter, *Opening Up to Open Source*, 6 RICH. J.L. & TECH. 24 (Spring 2000)
<<http://www.richmond.edu/jolt/v6i5/article3.html>>. ^[**]

TABLE OF CONTENTS

I. INTRODUCTION

II. WHAT IS SOFTWARE?

A. *What is Open Source?*

B. *The Open Source Movement*

III. SOCIETY SHOULD EMBRACE OPEN SOURCE

A. *Who's in charge?*

B. *Transition Issues*

C. *Practical Benefits*

D. *Software Economics*

E. *Copyright*

1. *Copyright and Open Source*

2. *Length of Copyright Terms*

IV. CONTEMPORARY OPEN SOURCE LICENSING AND TRANSACTIONS

A. *Infringement*

1. Copyright Remedies

2. Determining Infringement

B. Licensee and Licensor

1. Derivative Works

2. Terms, Conditions, and Warranties

V. OPENING UP

A. The Problem

B. Market Solutions

C. Government Solutions

VI. CONCLUSION

I. INTRODUCTION

{1}The latest "revolution" in the software industry has nothing to do with breakthrough technology; the revolution is a rethinking of how software technology is held, developed, and distributed. The revolution is called "open source," although it has also been called "freeware," and "copyleft." Each term generically describes the movement, yet implies wildly different ideas to the developers, distributors, and users inside the open source community. Open source is not a company, but rather, a community; projects are established and programmers communicate and contribute software building blocks to each other via the Internet. When a software program is completed by this method it is then offered to the public over the Internet, sometimes free of charge, but always free of the use restrictions common to most software.

{2}Open source products are growing in popularity. Linux, the open source operating system, captured 17.2% of the operating system market in 1998.[1]Forecasters predict that its popularity will grow faster than all other operating environments through 2003.[2] Apache, the open source webpage software, is now "the most popular HTTP server in use." [3] Although far from displacing its monolithic corporate competitors (e.g., Microsoft's operating system market share in 1994 was 90%),[4] its rapid growth calls for some analysis of open source's legal and policy implications, since its growth calls into question software copyright and licensing policies. Ironically, open source caught on slowly at first, because it was considered to be a product of academics and hobbyist programmers, and it was thought to be technically inferior to proprietary software. But, as the technical issues subsided, copyright, licensing, and warranty issues have arisen in their place. This article will discuss who, among open source programmers, has standing under copyright law to sue for infringement. It will introduce the interesting licensing issues which open source creates, and it will compare the warranties provided in an open source license to the warranties in a proprietary license.

{3}This paper is structured to address several purposes in its discussion of the open source movement. First, Part III will discuss on a broad level how society benefits from a software development process like open source, and how open source affects copyright and traditional notions of software piracy and reuse. Part III will also review solutions that open source offers and outline the problems that may occur as the open source model continues to unfold. Part IV then considers the contemporary objections to open source products, and

will demonstrate that copyright, licensing, and warranties - the legal issues in software - basically make purchasers no worse off under an open source environment than a proprietary paradigm. As such, businesses and consumers should not shy away from open source products. Part V crystallizes the underlying issue in this paper: open source has many beneficial aspects, but still, the market has been slow to accept it. The article concludes with possible solutions to the problem of slow acceptance, as the author asserts that both the market and government can take steps to foster the growth of open source. In the end, the reader will recognize that the traditional objections to open source software are fairly minimal. While open source may not reach revolutionary status, it opens the door to positive changes regarding intellectual property rights and licensing.

{4}To understand the issues presented in this article, one must consider some of software's origins. We begin with a brief history of open source software, as compared to regular, closed source (i.e., proprietary) software.

II. WHAT IS SOFTWARE?

{5}Originally, software was treated as a service. It was viewed simply as the labor component of a computer sales transaction. Purchasers would buy the computer, and the computer company would program it for them. Computer engineers commonly gave away software because it was the hardware that brought in the money.

[5] Initially, there was very little software available and "researchers typically swapped programs, embellishing one another's work without much attention to taking credit or nailing down commercial rights."

[6]

{6}In the late 1960's and 1970's, developers who were writing specialized software for particular clients wanted to protect their works. The "developers retained ownership of the software and 'licensed' the software to customers." [7] The licensing concept, derived from property law, basically grants permission to enter or use another's property. [8] The developers relied on property law because intellectual property is a "product of the human intellect that [has] economic value." [9] State trade secret and contract law were used, because patent and copyright at that time did not specifically cover software. [10] Software was still in its infancy and it was on the copyright statute's list of copyrightable items. Software became increasingly property-like, as it became increasingly available. Eventually, in 1976, after much deliberation, Congress applied copyright law to software, thereby strengthening the enforceability of the licenses. [11]

A. What is Open Source?

{7}In a way, open source is the re-emergence of the original software distribution model. Open source software differs in two primary ways from proprietary software. First, users receive both the source code and the compiled form, otherwise known as the object code. [12] The source code is the software's "blueprint," written in programming language (e.g., C, Pascal, or FORTRAN); whereas, object code is the series of 0's and 1's, also called machine-readable code. [13] The compiled form is "machine language" that works directly with the computer and is never seen by the user. [14] Proprietary software vendors generally sell only the compiled form. However, having the source code "open" allows basically anyone to read and alter the program or build a new derivative program. [15]

{8}Second, open source is licensed under a special "public" license such as the General Public License ("GPL"). [16] The GPL is issued by various open source software distributors. It comes in the form of the customary software mass market license. [17] The GPL is transferred by the consumer opening the box of software, or by a mouse click if the software is downloaded over the Internet. [18] The terms of the GPL encourage users to use the source code to make improvements, write new programs, and communicate all improvements and changes back to the original developer. [19] This approach continuously improves the program. The basic restriction of the GPL is that derivative products are distributed with the source code. In contrast to the GPL, a proprietary software license restricts licensees to use-only, and changes or

enhancements are prohibited.[20] Additionally, proprietary software is usually licensed for use on a limited number of computers.[21] The licensor reserves the right to make any improvements or changes.

{9}Open source developers claim both technical and social reasons for keeping the source code "open." From a technical standpoint, open source leads to a better product. With more eyes on the code, errors can be detected and corrected much more quickly than with traditional proprietary software. The social reason is that software should be free, meaning uninhibited and unrestricted, rather than without a cost or charge, which is the case with free speech. Used in this context, the term "[f]ree software" refers to the users' freedom to run, copy, distribute, study, change and improve the software" without restrictions or prohibitions.[22] Software pervades modern society; it can be found in almost every product. So naturally, if only a few people control software, their power increases and restricts users' freedom.

{10}There are valid arguments, however, for the proposition that software should not be free. These arguments include economic theories that making free software reduces incentives to software producers to create good products. The arguments against free software also stem from the basic business concerns regarding returns on investment. The question whether free software is a good policy can only be answered after a more full discussion.

B. The Open Source Movement

{11}The Open Source movement began at the time software developers started to sell software under licenses that restricted use and disallowed changes. Most consider the beginning was when Richard Stallman became dissatisfied with the way developers restrictively licensed software and decided to write "a complete UNIX-compatible software system," that he named GNU.[23] Stallman, a former computer programmer and researcher at the MIT Artificial Intelligence Lab, chose UNIX because it is portable, flexible, and a powerful multi-tasking operating system. Regarding his GNU system, he wanted to "give it away free to everyone who can use it." [24] Stallman asked "manufacturers for donations of machines and money . . . [and] individuals for donations of programs and work." [25] As more programmers became involved, Stallman issued the *GNU Manifesto* as a way to better explain the project and his concept of free software:

I consider that the golden rule requires that if I like a program I must share it with other people who like it. Software sellers want to divide the users and conquer them, making each user agree not to share with others. I refuse to break solidarity with other users in this way.[26]

{12}GNU gradually gained contributors, mainly in the form of academics and hobbyist programmers. Progress on the GNU project proceeded slower than the proprietary software development environment, but "[b]y the 1990s, [the GNU team] had either found or written all the major components except one --the kernel." [27] The kernel is "the fundamental part of a program. . . . [i]t is the part of the operating system that is closest to the machine . . . [it] activate[s] the hardware directly or interface[s] to another software layer that drives the hardware." [28] This problem was solved by Linus Torvalds' development of Linux, which is a free kernel.[29] With the contribution of Torvalds' kernel, GNU acquired a fully-functioning, UNIX compatible, operating system.

{13}Like Stallman, Torvalds also performed his first programming in an academic environment. Torvalds developed the Linux kernel, while he was a student teaching assistant at the University of Helsinki.[30] Desiring a stable operating system, he wrote the kernel "as a teaching aide for how such things should work." [31] Torvalds' friend suggested he name the kernel Linux- "a marriage of 'Linus' and 'UNIX.'" [32] Although Torvalds only developed the final kernel, the name LINUX stuck to the entire operating system.

{14}Early into the project, Stallman "decided to adapt and use existing pieces of free software wherever that was possible." [33] Initially Stallman employed TeX as the principal text formatter.[34] A few years later, he

decided to use the X Window System, rather than writing another window system for GNU.[35] X Window "is a windowing system . . .which runs under UNIX and all major operating systems." [36] X Window preceded the graphical user interface in the Microsoft Windows system. As a consequence of this decision, the GNU system differs from the collection of all GNU software.[37] Thus, many different free software applications compose the GNU system.[38]

{15}Stallman faced his first distribution dilemma in 1985 when his GNU Emacs program gained popularity. [39] UNIX programmers were clamoring to get a copy from Stallman, but Stallman was unemployed and "was looking for ways to make money from free software." [40] To generate income, Stallman "announced that [he] would mail a tape to whoever wanted one, for a fee of \$150. In this way, [he] started a free software distribution business, the precursor of the companies that today distribute entire Linux-based GNU systems." [41]

{16}As GNU's popularity grew, primarily among programmers and academics, Stallman and others founded the Free Software Foundation ("FSF"), a tax-exempt entity designed to promote free software development. [42] The FSF assumed control over the Emacs tape distribution business and later expanded the distribution business by adding other free software, both GNU and non-GNU to the tape, and selling software manuals. [43]

{17}As free technology improved and the community of GNU users and contributors grew, the term "open source" surfaced. Some members of the community decided to stop using the term "free software," substituting the term "open source software" in its place. The rationale for substituting the terms was simply to avoid the confusion of the word 'free' with 'gratis.' Other community members, however, wished to "set aside the spirit of principle that had motivated the free software movement and ... GNU project, and to appeal instead to executives and business users...." [44] Clearly, from this illustration of the dichotomy of perceptions about the open source movement, the terms 'free software' and 'open source' "describe the same category of software, more or less, but say different things about the software, and about values." [45]

{18}While Stallman is typically credited with starting the movement, GNU is not the only open source software. During the same time period, various developers and institutions also wrote and distributed under "open" licenses.[46] There are actually several open licenses with similar terms including licenses developed by Berkeley ("BSD"),[47] M.I.T. ("X Consortium"),[48] Debian,[49] and others.

III. SOCIETY SHOULD EMBRACE OPEN SOURCE

{19}This section highlights the policy issues at a macro-level regarding the following concerns: 1) the implications arising from a system reliant on freelance contributors; 2) the transition issues which arise when moving from proprietary to open source as the standard of software distribution; 3) the practical benefits of an open source standard of development; and 4) the economic benefits brought by open source.

A. Who's in Charge?

{20}The first contention against open source asserts that good software cannot result from an inconsistent development process that haphazardly creates products through the efforts of freelance contributors. In 1976, Bill Gates published "An Open Letter to Hobbyists," wherein he "accuse[d] hobbyists of stealing software and thus preventing 'good software from being written.'" [50] Gates wrote in that letter, somewhat prophetically, that "[n]othing would please me more than being able to hire ten programmers and deluge the hobby market with good software." [51]

{21}Gates has certainly deluged the world with good software, but is the Microsoft proprietary licensing scheme required to produce good software? Microsoft itself answered this question when it declared in an

infamous internal memorandum released in fall 1998, that open source systems were of a "commercial quality" that posed a "direct short-term revenue and platform threat."^[52] Although some claim that this memorandum was leaked purposely to the press so as to undermine the mounting antitrust evidence against Microsoft,^[53] it is evident from the Linux gain in market share that the freelance development system creates products of similar quality.

{22}A second claim contends that open source cannot exist without proprietary software vendors.^[54] Open source programming has been funded, to an unknown and indirect degree, by proprietary software vendors.^[55] These vendors employ programmers who, unbeknownst to the employer, work on open source projects, rather than their own assignments.^[56] If those programmers had to rely on a paycheck strictly from their open source contributions, open source probably would have ended. Undoubtedly, the financial incentives to write software are structured differently under open source.^[57] Yet there is evidence suggesting that open source returns may be sufficient now to promote creativity. Shares of Red Hat stock tripled on the first day of its initial public offering.^[58] VA Linux went public next, with shares soaring more than 600%.^[59] Several other open source companies are profitable now and preparing to go public.^[60] Although their combined net worth by no means rivals that of Microsoft, this evidence suggests that there is indeed a growing financial base to support open source.

{23}A third contention is that open source is, by necessity, a small system and that growth will cripple the movement. Assuming developers continue contributing open source code, "the distributed development model may ultimately fail. 'As time goes on, it gets harder for key developers to communicate because of the enormous noise on the mailing lists . . . Even if the development process continues to feed new improvements into Linux, this in itself may be ultimately destructive."^[61] Communication problems within the community are certainly a growing pain that open source must endure, but these problems are not extreme enough to prove destructive. If communication among freelancers was a problem, the journal and magazine publishing businesses would have collapsed long ago.

{24}One of the greatest strengths of Linux "has been its leanness, but the temptation to add more and more features can end up causing software bloat - huge programs trying to do too many marginal things."^[62] The concern with software bloat is no different for open source than it is with proprietary software. Under open source, however, any programmer has the freedom to remove the marginal features from the program, and to add any new features that the programmer deems appropriate.

{25}A fourth contention is that open source, like the Internet, will outgrow its freedom. The beginning of open source, just like the beginning of the Internet, was based on a more altruistic system. Both operated under a gift economy;^[63] therefore, incentives were not purely financial. Now, many people are making big money on the Internet, while only a few are making money from open source. There is little evidence that those reaping the open source benefits are sharing it with their programming contributors.^[64] This situation may cause the programming process to cease because contributors who were happy to help "the community" may not want to help "the company." For instance, hoping perhaps to keep the community happy, Red Hat reserved shares at its IPO price for the "friends of Red Hat" - the open source developers.^[65] In addition, Red Hat informed outside investors of the importance of this cooperative community: "software is created through the collaborative efforts of large communities of independent developers. Developers work alone or in groups to write code, make the code available over the Internet, solicit feedback on it from other developers, and then modify and share it with others for general use."^[66] Whether open source becomes less "free" in the future does not mean that it should not be embraced in the present. It is a viable alternative to proprietary software that, at least for now, allows users to become programmers.

B. Transition Issues

{26}Many companies, including Microsoft, have given products away free of charge to get a foothold in the

market.[67] However, the key transition problem concerning open code involves proprietary software vendors giving away source code to move their products into the open source market. The following discussion presents short term problems that will occur as open source continues growing and more former proprietary vendors shift to the open source model.

{27}The first shift is to sell proprietary products that operate on an open source operating system. Apple, Corel, IBM, Sun, and others have created, or are in the process of creating, applications to run on the Linux operating system.[68] The next step, which Corel is taking with its WordPerfect application, is to convert a proprietary application to an open source application. However, Corel committed the ultimate *faux pas* when it did not use an open license for a pre-release test version of the software, and thereby limited its distribution to only a few testers.[69] Corel is apparently still operating in the proprietary paradigm because the reason it did not open the license "is that [it does] need to protect [its] name and that [it has] to make sure that what [it] release[s] is good." [70] In the open source realm, products are released so the entire community can work and improve upon them. Keeping the source code a secret to be shared among just a few testers is contrary to the open source ideal.

{28}Netscape had similar problems when it open source edits Internet browser, Navigator, in 1998.[71] Netscape initially faced problems as "the monolithic architecture of its starting point (Netscape Navigator) caused severe delays in releasing usable code, and thus failed to attract developers." [72] Many software companies have a type of monolithic organizational architecture that hinders the movement to open source from the base level of the code itself. Usually, code is written by a small team of programmers who intimately know the code and the project, so that there is no need for documentation of potential bugs. A second level is the actual management of the code. Code is released slowly for a number of reasons. The company management, for example, may want to keep the potentially valuable components of a software application in-house, while allowing the less useful parts to trickle out to the open source community. Each part of the code is carefully reviewed for its profit potential before release to the public. Another management issue is that the company programmers may want to make some improvements to the code, knowing that their names will be associated with the pieces they wrote.

{29}Unlike a corporate software developer, individual open source contributors choose the projects they develop.[73] Consequently, progress can be quite slow on less popular programs. This fact is frustrating to those in the corporate mold, but in reality, it is a type of first-level test market. If a group of programmers is disinterested in developing a product, their disinterest may mean that the product is unlikely to fare well in the market. This approach allows programmers to develop beneficial products that may not have the cash return value that proprietary developers require before starting a project. Some may contend that this system hampers development because programmers can band together and boycott the development of certain products. However, the reality is that programmers from around the world contribute to open source development, and it is unlikely that their loose coalition would ever rise to the level of a conspiracy.

{30}At the 1999 Linuxworld Convention, open source guru, Eric S. Raymond, spoke of several conceptual business models that could work well for open source.[74] Raymond heads the Open Source Initiative, which acts as a missionary for the open source movement in general. The business models are, as Raymond characterized and coined them: 1) market positioner/loss leader; 2) widget frosting; 3) give away recipe/open restaurant; 4) accessorizing; 5) free the future, sell the present; 6) free the software/sell the brand; and 7) free the software/sell the content.[75]

{31}A market positioner or loss leader "use[s] open-source software to create or maintain a market position for proprietary software that generates a direct revenue stream." [76] Open source client software can enable sales of server software, or generate advertising revenue on an Internet portal.[77] Netscape Communications, Inc. seemed to follow this strategy by releasing the open source Mozilla browser in early 1998. In fact, "[b]y opening up the widely popular Netscape browser, Netscape effectively denied Microsoft

the possibility of a browser monopoly." [78]

{32} "Widget frosting" is a model for hardware manufacturers because they view software as overhead, rather than a profit center. [79] Market forces have compelled hardware manufacturers to write and maintain software, such as device drivers. If they used open source software, their maintenance and writing costs could be placed on the open source community. As a result, "the vendor gains ... a dramatically larger developer pool, more rapid and flexible response to customer needs, and better reliability through peer review." [80] At least one manufacturer has used this method: "Apple Computer [employed this model when] they open-sourced 'Darwin,' the core of their MacOSX server operating system." [81]

{33} "Giving away the recipe and opening a restaurant" means "selling the value added by assembling and testing a running operating system that is warranted (if only implicitly) to be merchantable and plug-compatible with other operating systems carrying the same brand." [82] This is what Red Hat and other Linux distributors do. These companies also generate revenue by selling service to install the software as well as support service contracts.

{34} "Accessorizing" means "sell[ing] accessories for open source software." [83] This activity could equate to, "[a]t the low end, mugs and T-shirts; at the high end, professionally-edited and produced documentation." [84] For example, "O'Reilly Associates, publishers of many excellent references [sic] volumes on open-source software, is a good example of an accessorizing company." [85] Obviously, accessorizing is not limited to the open source realm. Independent writers, for instance, have published and sold tutorials and manuals for all kinds of software applications. Under open source, however, the writer has the entire source code available and can write an intricately detailed manual.

{35} "Freeing the future and selling the present" involves a play on licensing terms. Under this model, the software is released under a closed license, but the license includes an expiration date on the closure provisions. For example, a license may be written which "permits free redistribution, forbids commercial use without fee, and guarantees that the software comes under GPL terms a year after release or if the vendor folds." [86] This method is in line with the idea that copyright creates market inefficiencies, as was discussed previously in this article. The software developer has a short term monopoly in which he is the sole profit recipient. Upon expiration of the term, others have the right to alter the software and to create derivative products. This method comes closer to striking the balance between the needs of the individual software creator and the public's needs than other models.

{36} "Freeing the software, selling the content" means giving away software that is only a means to something else. Many Internet Service Providers ("ISPs") use this method, when they mail free CDs to consumers so that when they install the free software, they can pay the ISP for the information it receives through its portal. The ISP would be better off by open sourcing its dial-up software in the same way hardware manufacturers benefit under the widget frosting idea discussed above. Furthermore, it is Raymond's observation that "[t]he value is neither in the client software nor the server[,] but in providing objectively reliable information" which is why Raymond believes that America Online should move its client software to open source. [87]

{37} Raymond's last idea has future implications. "Free the Software, Sell the Brand," means that the developer retains a test suite or a set of compatibility criteria, so that when users implement an open source application that has been altered from its original specification, the developer checks out the altered application, and if it meets the compatibility criteria, the user is sold a brand certifying that their implementation of the technology is compatible with all others wearing the brand.

{38} The idea of branding may become very important to the future growth and existence of open source software, because it addresses the lurking concern of industry standards regulation. There is no single source

to answer questions or to establish product standards. Presumably at a future date, Red Hat Linux could be very different from Caldera Open Linux. Is this really a problem? Not necessarily. Recall that when PCs were first available, there were several choices for operating systems and several choices for each type of application. Standardization occurred in an evolutionary manner. Eventually, the choices were limited to a relatively few monolithic software companies. For many, this reduction in choice was a relief because it also reduced the uncertainty that the software would be supported in the future. If the market pushes for standardization of open source products as strongly as it has in other areas of computing, open source will fail without some level of compatibility branding. Even if it fails, with the source code freely available, chances are much higher that some software engineer can be hired to support it.

{39}Even by applying Raymond's business models, not all developers will successfully convert to the open source method. We have already noted that, while Netscape's Mozilla release may have helped Netscape successfully avoid a Microsoft browser monopoly, "the Mozilla project has little to show for in the way of usable code."[\[88\]](#)

C. Practical Benefits

{40}For our computerized world to continue turning, our computers must be compatible. The quest for compatibility led to industry standards in architecture on the hardware side and jousts for market domination on the software side. For example, Microsoft's MS-DOS (and now Windows) was the de-facto operating system standard, and for years, there was no real rival in the PC market. Consumers came to know that they could purchase and install any software that operated on DOS. This market domination removed consumer's compatibility concerns, but this domination ultimately triggered an antitrust action against Microsoft.[\[89\]](#) Open source avoids antitrust and monopoly issues because no single entity owns the code. Compatibility problems are also tempered because each contributor wants to ensure that his contribution will succeed.

{41}In addition, open source may be a force for innovation in the hardware market. Innovations in hardware have remained relatively static, as compared to software.[\[90\]](#) Certain hardware and software developers have formed strategic alliances, but smaller developers are normally left to themselves. Relying on industry standard open source code, hardware developers can focus on improving their own products, rather than on making sure the hardware will work with the various software applications.[\[91\]](#) Hardware developers like the fact that open source is royalty free and that it is completely configurable.[\[92\]](#)

{42}Open Source may encourage computer users to become more knowledgeable. Simply having the option to program one's own software "fix" is a start, even though most users are not programmers. Users have not been encouraged to learn how to write source code because as consumers, we have been taught that writing code is the job of software companies. Software companies encourage this thinking because it increases their economic security, but it has also caused a shortage of information technology professionals in the United States today. The technology worker shortage will eventually be replaced by a glut in the market. The previous oversupply of computer workers still affects the marketplace; for instance, older information technology professionals continue to find difficulty locating employment. Information technology employment cycles can, in part, be smoothed by a widespread open source standard. Open source allows individual programmers, unaffiliated with any software company, to practice the trade on a smaller scale, while still working full time in another occupation. The advantage is that the hobbyist programmer is getting valuable feedback from others in the open source community about his work. Without that feedback, the hobbyist has minimal value in the market. By keeping his skills honed, he is able to easily jump into programming full-time when the market gives him incentives to do so.

{43}The older worker problem originates from firms that have phased out older technology and its accouterments. As an illustration, recall that no one seemed to need COBOL programmers when new programming languages became available that worked better on the PC platform.[\[93\]](#) But with some

remedial training in the C programming language, and with source code freely available, these older workers may be able to work as freelance programmers for companies moving to open source platforms, or as contributors to one of many open source projects.

{44}Open source is good for the Internet. The growth of open source may even follow a pattern of growth similar to that of the Internet. There are three basic reasons for this predicted optimistic pattern of success for open source. First, as discussed earlier, open source and the Internet grew up together.[94] The software used to develop compatible protocols was open source from the beginning, so that everyone could have access to it. Therefore, the more the Internet grows, the more open source will grow. Second, the prevalence of open source software on the Internet will make the Internet very difficult to regulate.[95] However, regardless of one's personal opinions about Internet regulation, having the Internet technologically incapable of regulation for a time creates an exciting experimental hypothesis - can the market actually regulate itself to the benefit of all Internet users? If the market can provide solutions, government may not have to expend resources in regulating the Internet.

{45}Third, the Internet has changed the way software is priced and valued. Software that operates the Internet is often given away free of charge or is paid for by advertisers.[96] Companies have recognized that it is the content on the Internet that has the value, and not necessarily the software tools used to access it. By analogy, open source diverts the value usually found in the price of software into the price for content, maintenance, warranties, etc.[97] Price shifting, in this case, creates a truer economic picture of consumer value. Consumers are not just purchasing software; instead, they are purchasing access to the content of the Internet. With maintenance and warranties, they are insuring that access to the Internet and their other software applications will function properly.

D. Software Economics

{46}The basic economic model assumes scarcity of product, and greater scarcity generally means greater value. This assumption causes problems for software because it is only as scarce as vendors and consumers make it. The whole world can collude in purchasing one licensed copy if the first purchaser gives copies to everyone else. Proprietary software vendors call this act "software piracy" - stealing.[98] Others prefer the term software "reuse." [99] The economic term is "free rider" - meaning a person who takes advantage of someone else's benefits and activities without paying.[100] Extensive free riding can cause a market failure. Allowing outright copying of software results in a market price around the cost of duplication,[101] which completely ignores development and marketing costs. At that low price, there is little incentive for developers to supply new or better software. Overall, an undesirable result develops from this chain of events.

{47}Does the current method of protecting software avoid the free rider problem? Not conclusively. The theft of copyrighted software in 1998 amounted to nearly \$11 billion,[102] which is one-third or more of the value of the software sold by U.S. producers.[103] The free rider problem inherent in proprietary software greatly diminishes in an open source model. Free riding is a non-issue when the developer expects no (or minimal) payment in return for his contribution to the software.

{48}The question begging to be answered is how open source can convince software developers to forego the traditional forms of payment for their labor. This question is answered in part by Raymond's business models and by the recent success of open source companies such as Red Hat and VA Linux. The part of the question that goes unanswered is whether this method of open source licensing will continue to gain popularity in the market over the older, more established proprietary models.

{49}The current method of coupling copyright with restrictive licensing terms leads to economically inefficient results because software is "expensive to create and companies can save costs by reusing pre-existing work." [104] Open source developers continue to build on one another's works, whether by

contributing a simple bug fix, or by creating a compatible application. The word "application" means a function of the computer, such as word processing, spreadsheets, games, etc. The advantage of free software programmers is that they "don't have to solve the same problems over and over. They keep improving on the work that came before, like the scientific method."[\[105\]](#)

{50}The assumption behind copyright law is that protection is necessary to promote innovation, which in turn, makes restriction economically efficient. Innovators can also choose not to take advantage of copyright protections, as open source developers have chosen to work without the mini-monopoly created for them by copyright law. The open source business models, introduced by Eric Raymond, do not require software developers to hide their software secrets to make money. The same business models help open source avoid piracy issues. The free rider problem of proprietary software is as much caused by the miscategorization of software as a "manufactured good" as it is by "pirates." The public has "a strong tendency to assume that software has the value characteristics of a typical manufactured good."[\[106\]](#) The general perception is that, "1) [m]ost developer time is paid for by sale value [and] 2) [t]he sale value of software is proportional to its development cost."[\[107\]](#) According to open source advocates both assumptions are false. Unofficial open source "surveys" indicate that 95% of software is written in-house specifically to meet in-house business needs.[\[108\]](#) One open source vendor states that his "usual development customer is someone who needs a problem solved, but who is not in the software business."[\[109\]](#)

{51}There is a concern that making source code freely available could increase piracy and reduce the value of the software to the cost of duplication. However, by reducing the amount of value artificially infused into the software component of the product and by placing value in other items or services packaged for sale (i.e., support, warranties, maintenance, etc.), simple copying may not reduce value to market crashing proportions. Consumers often buy a software application that includes in its price some form of limited technical support. The technical support may or may not ever be used, and it may not meet the value paid by the consumer. The value of the software product is not just in what is on the diskette or CD.

{52}It may sound rather unfriendly for Red Hat to sell a software suite without support for a price of \$29.95,[\[110\]](#) while it sells the same product coupled with maintenance and support for \$79.95.[\[111\]](#) Even the least wary consumer can be motivated by the apparent economic incentives for poor workmanship as a sound rationale for purchasing technical support. However, for those who are comfortable finding and correcting software glitches on their own, this pricing scheme is quite a bargain. While not every consumer may want to debug his own software, at least this approach offers consumers a choice. Microsoft prices Windows to include support, but there is no way to bargain for a lower price, if the individual consumer does not desire support.

{53}Furthermore, a software product's value does not always match its price. Red Hat gives away Linux over the Internet.[\[112\]](#) The \$29.95 software product consists of Linux and an office suite that includes a word processor, spreadsheet, and presentation program.[\[113\]](#) In comparison, Microsoft Windows 98 operating system costs \$96.82,[\[114\]](#) and its office suite comprised of Word, Excel, and PowerPoint Presentations costs \$329.90.[\[115\]](#) In addition, it must be noted that if a consumer has a choice between two goods that perform substantially the same function, although one is technically superior, the consumer will often choose the cheaper good. This theory has often been used to explain why the VHS format overtook the Beta-Max format of video recorders. Leaving out the technical superiority question, open source is currently a second best alternative, and it is gaining momentum.

E. Copyright

{54}One of the first acts passed by Congress was Copyright Act of 1790.[\[116\]](#) The Copyright Act underwent a major overhaul in 1976, in order to, among other things, broaden the scope of copyright to include "all writings," rather than just books.[\[117\]](#) The Copyright Act of 1976[\[118\]](#) presents Congress' most recent major

reform of the Act "which [with some modifications] governs most works today."^[119] But, it was not until 1980 that the term "computer program" made it on the list of copyrightable items.^[120] The U.S. Constitution states the original purpose of patent and copyright: "[t]o promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."^[121]

{55} Patent protection is available, but difficult to obtain, so most software developers copyright their products. In short, patents protect " any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof. . . ."^[122] Copyrights protect the expression of ideas. It is often difficult to patent software because most software is simply a new expression of an existing process (i.e., accounting software is just a computerized expression of pen and paper accounting). Copyright protection is granted to the creators of "original works of authorship."^[123] The subject matter of the Copyright Act includes literary, dramatic, musical, and pictorial works.^[124] Among the rights that accompany copyright protection are the right to reproduce the work, prepare derivative works based upon the work, make copies of the work, perform the work publicly, and display the work publicly.^[125]

1. Copyright and Open Source

{56} The constitutional purpose for granting copyright privileges is to further science and the useful arts.^[126] In other words, the benefits of copyright law should flow to society, and not strictly to the author. Economists recently have argued along the same lines regarding software.^[127]

{57} The proprietary software model restricts users from altering or copying software. As such, software developers have to work in a vacuum, unable to avoid the mistakes others have made before them. This model only benefits the software developer who comes out on top. This author maintains that software copyrights should be construed in a way that increases the returns flowing to society. If software developers could have the opportunity to build on each other's ideas, rather than duplicate each other's efforts, the benefits would return to society, and not just to the developer. The economic argument favors relaxing the copyright laws and allowing some reuse.^[128] The "economic goal of copyright law is to balance an author's incentive to create with his or her ability to *build on prior work* in order to maximize social wealth."^[129] Open source allows reuse, but it does so only by foregoing some benefits of copyright law.

{58} Open source developers copyright their products.^[130] If they simply put their creations in the public domain, our current law would end open sources. Software left unrestricted in the public domain (i.e., placed on the Internet for unrestricted use) allows subsequent developers to legally take it, alter it, and sell it under a restricted use license. To avoid this loss, the software product is copyrighted to give the original writer control over its use, but the open source developer places different restrictions on the product than a proprietary developer does. The key open source restrictions are: 1) the software must be redistributed freely; 2) the distribution must include the source code; 3) derived works are allowed; and 4) the author's source code must retain its integrity.^[131] Thus it can be said that copyright restrictions on open source software ensure that it remains free.

2. Length of Copyright Terms

{59} The bundle of rights granted by a copyright remains exclusive to the original author "for a term consisting of the life of the author and seventy years after the author's death."^[132] In comparison to the copyright term, the useful economic life of a software product is much shorter. For example, from 1981 to 1993, Microsoft released six major MS-DOS products.^[133] Windows, which first retailed in 1985, is currently in its sixth major iteration.^[134] Roughly every two years, the previous version becomes obsolete.^[135] Generally, copyright owners make money by selling permission (i.e., licenses) to reproduce, copy and make derivative works from the original products; whereas, owners of software copyrights tend to sell

licenses for use only. Open source software licenses allow use, reproduction, or creation of derivative products. This licensing method puts control in the hands of the users regarding the release of new upgrades, which features are added, and which bugs are fixed. In a way, proprietary vendors use the terms of the license against consumers.

III. CONTEMPORARY OPEN SOURCE LICENSING AND TRANSACTIONS

{60} Corporate purchasing, information technology ("IT") and legal departments have their own concerns about open source software. Corporate purchasing departments worry that open source vendors offer no warranties. IT professionals worry that the product will not be supported long enough to recoup the corporate investment. Legal departments worry that the company will not own the software it creates from open source and that someone else will have rights in the company software. Legal fees to defend a copyright infringement suit averaged \$249,000 in 1997.^[136] Consumers might also be concerned about these three issues. We will consider these issues in the remainder of this article, beginning with the legal concerns.

A. Infringement

{61} While open source licensees may worry about the number of people who have rights in their software, there should be little concern manifested overall. The open source copyright holder has fewer rights to sue licensees than proprietary copyright holders because literally all open source copyright rights are granted to the licensee through the GPL.^[137] The GPL grants licensees' rights to modify their copies of programs, forming works based on the programs, and to copy and distribute such modifications.^[138] GPL licensees can distribute the modified programs as long as they also meet the following three restrictions. First, they must "cause the modified files to carry prominent notices stating that [they] changed the files and the date of any change."^[139] Second, "any work that [licensees] distribute or publish, . . . [must] be licensed as a whole at no charge to all third parties under the terms of [the GPL]."^[140] Third, the modified work must "print or display an announcement including an appropriate copyright notice and a notice that there is no warranty [or else, saying that [they] provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License."^[141]

{62} There is only one restriction on distribution among these three alternatives. Specifically, the distribution must be accompanied by "the complete corresponding machine-readable source code," or, "a written offer, valid for at least three years, to give any third party. . . a complete machine-readable copy of the corresponding source code," or, accompany it with the information "received as to the offer to distribute corresponding source code."^[142] It is important to note that "[t]his alternative is allowed only for noncommercial distribution. . . ."^[143]

{63} The GPL restriction that derivative works "must . . . be distributed under the same terms as the license of the original software[.]"^[144] causes concern for those who write software for in-house use. For example, consider the petroleum company that writes its own software to calculate oil well capacity. If the engineer builds upon an existing, copyrighted open source product to create the application, is his product a derivative product subject to the open-distribution restrictions of the GPL? Yes, there is no doubt that it is a derivative product. Next, we must ask, will he have to make his product open to his competitors to free ride? What if our petroleum engineer decided to sell the derived product to other oil companies under a proprietary license? The answer is readily apparent, as this is clearly a license violation, and it is also a copyright rights violation.

{64} What if the engineer, instead of creating a derivative work based on an open source product, used a GNU compiler or library to help create the oil well software? A compiler is a software tool that translates the source code into object code. The object code version of the software created through the use of these tools may constitute a derivative work, in this case, a translation. The law is currently not entirely settled on this question.

{65}The open source community has responded to this problem by creating a new license for libraries and compilers, and it is called the Lesser General Public License ("LGPL").^[145] LGPL is intended "to preserve the modifiability and redistributability of LGPL software without encumbering works that '[contain] no derivative of any portion of the Library' with the LGPL."^[146] However, even though the intent was to avoid the derivative work problem, the LGPL does not solve the problem. Rather, "[t]he LGPL gets more fuzzy as the distinction between the proprietary work and the library are blurred. For example, an extension library based upon the library may be a derivative work, or it may not be, depending on interpretation."^[147] The intent of the LGPL is to regard programs compiled by the GNU compiler as non-derived programs, but the interpretation may ultimately be left to the court if the dispute ever arises.

{66}To date, no open source contributor has sued a subsequent licensee for copyright infringement and neither have any of the open source vendors.^[148] A suit may arise, however, under a breach of license claim, as was already discussed previously in the portion of this paper addressing the compiler issue. Of course, obviously a licensee could sell an open source derived product under a proprietary license. No court has ruled on the validity of an open source copyright or determined liability or damages based on infringement of an open source license. One legal writer contends that the GPL is a valid enforceable license, stemming from shrinkwrap licenses and shareware licenses.^[149] The oldest commercial open source developer, Cygnus Solutions, claims the GPL's "terms are specific and sufficiently narrow[,] that it is viewed as being as valid as any other software license."^[150]

{67}Assuming our engineer infringes the copyright, who has standing to bring suit? According to the Copyright Act, only the legal or beneficial owner of an exclusive right under the copyright may sue.^[151] The legal owner is the person holding the copyright.^[152] Beneficial owners are those who have "parted with legal title to the copyright in exchange for percentage royalties based on sales or license fees."^[153] One court has held that, "[t]o bring an infringement action, a plaintiff must be the owner of a copyright, its assignee, or an exclusive licensee."^[154] GNU, and most other open source products, are distributed solely under a non-exclusive license.^[155] This licensing method would seem to limit the number of possible plaintiffs if the license or copyright is infringed.

{68}The first question in determining standing is whether the plaintiff developer is really a new owner of copyrightable files. Due to the teamwork nature of open source projects, most developers will be non-exclusive licensees creating a derivative work under the GPL. They are licensees because the only way to gain access to the program they are improving is by obtaining a license to it. There may be multiple owners of subparts of any given open source software application. For example, "the Linux kernel ... [is] a patchwork of copyrights. Each individual file may have its own copyright, or sometimes even multiple copyrights."^[156] The problem of standing results from the collaborative nature of open source development. There is no question that open source developers retain copyright rights in programs they create and release themselves.^[157]

{69}Once a plaintiff with standing decides to sue, it may be procedurally onerous to continue. The court may require the plaintiff to serve notice of the suit to other parties with an interest in the copyright. Once initiated, "[t]he court may require the joinder, and shall permit the intervention, of any person having or claiming an interest in the copyright."^[158] The list of parties can become rather long if each code contributor can be considered an interested and necessary party.

{70}Open source licensors seem to have been cut off from asserting copyright claims at all in the Ninth Circuit, which ruled that, a "'copyright owner who grants a nonexclusive license to use his copyrighted material waives his right to sue the licensee for copyright infringement' and can sue only for breach of contract."^[159] Open source licensors distribute solely under nonexclusive licenses. But limiting the copyright owner to breach of contract actions gives rise to other problems, such as invalidity for lack of consideration (free download from the Internet is common), and for lack of privity. Following the decision of

the Ninth Circuit, the open source developer appears to waive the right to sue for copyright infringement and loses any contract remedies.

{71} In the same case, *Sun Microsystems*, the Ninth Circuit went on to allow copyright remedies for a certain breach of license. If "a license is limited in scope and the licensee acts outside the scope, the licensor can bring an action for copyright infringement."[\[160\]](#) The GPL, by its terms, grants licensees all of the licensor's exclusive copyright rights. With such a broad grant of rights, is the GPL sufficiently limited in scope?

{72} As earlier discussed, the GPL grants licensee's all of the licensor's copyright rights. The only substantial right reserved by the GPL is that any derivative products made for distribution by the licensee must also be distributed in the open source format. Case law supports a finding that breach of the GPL amounts to copyright infringement.[\[161\]](#) Courts have found infringement by exceeding the scope of a license when the licensee used the software on hardware unauthorized by the license,[\[162\]](#) when the licensee allowed third party access to the licensed software,[\[163\]](#) and when the licensee made "innumerable copies" of the software for unauthorized purposes.[\[164\]](#) Each breach of the license agreement violated only one of many license restrictions.[\[165\]](#) By comparison, the GPL contains basically only one restriction, that further redistribution must be made under the same license.[\[166\]](#) This violation appears at least as serious as the three under which infringement was found.

{73} The previous cases seem to establish that the GPL is sufficiently limited in scope to support a copyright claim. The next question concerning the validity of the license is whether the limitation is enforceable. Limiting the terms of distribution could possibly be declared unenforceable by a court as contrary to the public policy.[\[167\]](#) The situation would arise when the alleged infringer asserts a defense of copyright misuse. A copyright misuse defense trumps anti-competitive restrictive clauses in software licenses. The dominance of the copyright misuse defense has been shown in cases where the license prohibits licensees from making software based on the basic idea underlying the licensed software,[\[168\]](#) and where licenses prohibit the production of any competing product "whether or not computer-based."[\[169\]](#)

{74} Does the restriction of the GPL on distribution rise to the level of these anti-competitive clauses? This author believes the answer is probably not, for the three following reasons. First, open source is not a monolithic competitor. Within the open source community, there are various competing companies selling almost identical software under the exact same license - the GPL.[\[170\]](#) Second, for every open source software application, there are myriad proprietary versions available that accomplish the same result, and that can be purchased under different license terms. Third, the open source distribution limitation is minuscule when compared to the restrictions found in proprietary licenses. The GPL restriction on distribution is not anti-competitive and should survive a copyright misuse attack. This means that, just like any other software licensor, open source licensors have copyright standing to sue for breach of license.

1. Copyright Remedies

{75} For what damages would the infringing engineer be liable? Copyright law establishes statutory damages for infringement up to \$150,000 or for reimbursement of the infringer's revenues.[\[171\]](#) Apportioning damages among all those with an interest in an open source product may prove difficult. For example, how can one value one developer's contribution against that of another? It would be like evaluating the relative values of a word processor's thesaurus and its spell checker. Splitting the damages evenly among all code contributors is also unfair. The difficulty of apportionment and the nonjoinder of some interested parties may decrease the damages awarded. Copyright also allows injunctive relief.[\[172\]](#) The developer who created a derivative work and distributed it under a proprietary license may be required to come into compliance with the license and to make the source code available.

2. Determining Infringement

{76}The difficulties with remedies aside, some parts of an infringement proceeding are much easier under open source. A substantial amount of judicial effort goes into determining *if* copying has occurred. Parties in several software cases have done extensive side-by-side comparisons of source code to prove the act of copying. Courts have found infringement when as little as fifty-six lines of code have matched between programs.[173] Parties often do not want to release their source code to the court in fear that it will somehow be leaked to the public. Because source code is freely available, the parties will be unconcerned about the litigation that often requires the release of their code.

B. Licensee and Licensor

1. Derivative Works

{77}Returning to our engineer who created a derivative work from an open source product, it is highly unlikely that he will be stuck with a license infringement suit. The GPL explicitly allows modifications and derivative works.[174] The open source license requires that derivative products be "distributed under the same terms as the license of the original software." [175] As to derivative works that are not distributed, there is no need to open the source to competitors.

{78}Our hypothetical engineer is a great concern for corporate legal departments. As the problem is defined, the engineer has done no wrong. If, however, the engineer acquired the original source code from which the derivative program was created, from a copyright infringer, the company will be in trouble. Open source software licenses provide no warranties, not even the warranty against infringement. In a situation such as this, the company may become involved in the complex litigation scenario contemplated above.

2. Terms, Conditions, and Warranties

{79}Assuming the license is valid, what about the warranties, terms, and conditions of the GPL for those who just want a reliable software product? Open source vendors tend not to provide warranties for their products, presumably because no single person or group is responsible.[176] But, this is nothing new for consumers. Proprietary vendors also tend not to provide warranties for their products. Comparing a Microsoft Windows 95 license agreement to the Free Software Foundation's GPL, there are some striking similarities. Foremost, both licenses disclaim all implied warranties.[177] Next, neither license guarantees that the product will work.[178]

{80}Microsoft allows use of a product on only one computer at a time;[179] whereas, the GPL allows unlimited use.[180]Microsoft disallows reverse engineering and disassembly to remove any component part, [181] while the GPL allows any alteration.[182]Microsoft warrants that the software will "perform substantially in accordance with the accompanying written materials for a period of ninety (90) days..."[,] [183] but beyond that claim, there are no warranties, either express or implied.[184] The GPL software, on the other hand, is "as is without warranty of any kind," [185] for any amount of time. Microsoft's only remedy is either, "return of the price paid" or "repair or replacement of the software" at the manufacturer's sole option.[186] The GPL grants no remedies.[187] If the license is any indication of the product's viability, Microsoft will guarantee that the software will work on one computer for ninety (90) days or they will send a replacement disk.[188] The FSF gives no guarantees, but allows users to do whatever they want with the software.[189] In one respect, buyers are a little better off under the GPL than they are under a proprietary license because at least they have the option to alter the code.

{81}Both proprietary and open source licenses disclaim consequential damages.[190] But, open source software security may give rise to consequential damages problems.[191] With operating system source code freely available, it may appear easier for malefactors to create and to load destructive viruses.[192] In reality, it is actually not any easier to create viruses in open source than it is in proprietary operating systems.[193]

Proprietary developers must provide in-depth details about the executable formats so outside software developers can write applications.[194] These details are enough to allow hackers to hook in and build a virus.[195] The upside in open source is that "there are more people watching for these hacks ... so they can be caught quicker,"[196] and the fix can be communicated to all users more quickly than in a traditional closed source situation. Open source facilitates quicker communication because once someone encounters a problem, the problem may be posted on the Internet. Then, users begin working together to fix the problem.[197] By comparison, some software companies will not publicly admit that a problem exists until they can deliver a fix.[198]

{82}To fix a virus in a web server, Microsoft's lead product manager Jason Garms says "[t]he absolute minimum [time] expected to fix [a virus] is two weeks." [199] In comparison, the turnaround time is drastically lessened in an open source environment, such as in a similar situation involving "the teardrop exploit, [a bug] that affected both Linux and Windows systems. It was patched under Linux in a matter of hours." [200] Linux has two advantages over Microsoft operating systems in the bug fixing area. First, the number of Linux users is still very small in comparison to the number of users of other systems, so it is easier to reach each individual user. Second, until recently, the great majority of Linux users were also Linux programmers or at least computer literates, who could quickly detail a problem and understand how to apply a fix. As Linux and other open source applications grow in popularity, these two advantages will likely diminish.

{83}For those who feel more secure when they can point to a corporate product manager to fix problems, closed source may "create a false sense of security." [201] An open source fix is not guaranteed in a license agreement, but it is "warranted," at least by the pride of the individual developer that wrote the program, essentially guaranteeing that the person will do all that is necessary to improve his product.[202] There are hundreds of viruses for Windows and only a handful, possibly under ten, for Linux.[203]

{84}There are legitimate needs for encryption and secrecy software to protect sensitive information on government computers, in banks, the military, and the like. Having encryption software open-sourced would seemingly defeat its very purpose. Strangely enough, experts consider open source encryption software technically superior to many proprietary products.[204]

{85}Corporate purchasers commonly want some kind of warranty that someone will support the software well into the future, especially if the software is an integral core from which other applications will be built. In one respect, open source solves this problem because the source code is available and the purchasing company can train someone in-house to maintain it. Current proprietary programs must tailor its features to the lowest common denominator. If a specific user needs a bug fixed or wants additional functionality there are two options: 1) ask the vendor to fix the bug or add the function; or 2) find a new vendor. With open source, there is always a third option - "fix it yourself." [205] If you cannot fix the problem yourself, there are several open source vendors who can and will provide a fix.

IV. OPENING UP

A. The Problem

{86}The preceding sections of this paper have introduced open source software and outlined some of the benefits that can result from its broad acceptance. The article has also presented a general problem open source software faces - the market has been slow to accept open source. The market has been slow to accept open source partly because of product concerns. Product concerns include product viability, technical issues, such as security and scalability, and technical support. Product concerns seem to be the first barrier to any new product and are certainly not endemic to open source. While product concerns are a key issue for the open source community, offering technical solutions are beyond the scope of this article.

{87}A second reason for slow acceptance is the user's contractual concern that they might sign a purchase contract with a company that did not create the product they are buying. Programmers may be uneasy developing open source products because their copyright rights are minimal. Open source software companies may be uneasy providing warranties of any kind for products they did not create. These contractual concerns are only beginning to be addressed. Even with some companies now offering warranties with open source products, some consumers are still choosing to go without the warranties. Along these lines, many purchasers simply refuse to open contractual discussions with a small company that may not be around in the future to support the product. This affects the development of open source because many open source companies that support open source products are still small and have short corporate histories.

{88}A third problem that may limit the growth of open source is product standardization. This problem stems from the free availability of source code. Programmers can add functionality, enhance, or simply alter the application, such that it can be sold as a different program or at least a newer version. The multiplicity of products and versions can result in incompatible systems and inconsistent products.

B. Market Solutions

{89}The assertion that the market provides a solution is a little inaccurate. What really happens is that an individual buyer or seller settles the issue in such a way that other buyers and sellers follow suit. In fact, the market has already started to work on the contractual concerns arising. Some users are used to going without warranties on their proprietary products, so the thought of not having a warranty for an open source product does not worry them. On the other hand, open source vendors increasingly offer support contracts and guarantees that the product will perform. Beyond this "give-and-take" relationship between users and purchasers, the market offers no further solution to contractual concerns.

{90}However, there are a few ways the market can attack the problem of standardization. First, open source companies can form a separate, non-profit entity to verify application compatibility. This could be similar to the non-profit, multiple listing services created by local real estate agencies. In the absence of cooperation among companies, a separate entity could step in to offer independent verification. Software escrow companies already offer independent source code verification for companies on an individual basis. This standardization or verification entity could operate as a type of brand for developers or as a certification for users. For a fee, the entity would test new products on criteria based on compatibility with other products, platforms, or devices. If the new product met the criteria, it would be given a brand that it passed. The brand may be similar to the Underwriters Laboratories ("UL")[\[206\]](#) listing for electronics. The entity could also operate on the user end as more of an information clearinghouse on compatibility and standards. The key to success is that the entity must establish consumer trust by remaining impartial and independent from vendors.

C. Government Solutions

{91}There are three main areas in which government can also enhance the adoption of open source. First, states can amend the Uniform Commercial Code ("UCC"), which governs the sale of goods, to reduce contractual concerns. Second, the government's vast computer resources systems can be operated on open source software. This software utilization could give open source the type of exposure that will reduce many product concerns. Third, judicial remedies can include requiring certain products to be licensed as open source. This requirement could potentially place more products into the open source market. A fourth area involves copyright issues. As previously discussed, open source already works well within existing copyright law. Congress could scale back many of the software copyright protections, so as to make all software open source, but doing so would limit the freedom currently enjoyed by other software developers. Open source, after all, is about freedom.

{92}State legislatures should adopt amendments to the UCC. The code is written by a non-profit entity

consisting of judges, law professors, and practicing lawyers, the National Conference of Commissioners on Uniform State Laws ("NCCUSL"). The NCCUSL has already promulgated an amendment to the UCC called the Uniform Computer Information Transactions Act ("UCITA"). The UCITA governs certain computer software and information transactions, but for the most part it does not apply to computer hardware, which are considered to be goods. No states have yet adopted this amendment draft.

{93}The scope of UCITA goes beyond open source software, but it can directly benefit open source software by codifying the legality of mass market licenses. The current draft advances the codification of mass market products. Some courts have already found the use of some mass market licenses legally binding, such as "shrinkwrap" licenses in which the buyer assents to the terms of the license by breaking the transparent shrinkwrap on the software box.[\[207\]](#) This codification moves a step closer to legal approval of binding "click-wrap" licenses. The GPL is a mass market, click-wrap license. That means that the buyer manifests assent to the terms of the license by virtue of clicking a button on a website. The unquestioned validity of mass market licenses is a necessity for open source because a large number of open source transactions occur between anonymous parties over the Internet.[\[208\]](#) But, the UCITA is far from being an open source booster. Depending on the language state legislatures adopt, the UCITA can as easily invalidate open source click-wrap licenses. The UCITA also poses other threats to open source software development regarding warranties and reverse engineering.

{94}The current draft builds new implied warranties into software licenses. Implied warranties of any kind are uniquely difficult for open source developers. Because many individual open source programmers receive little compensation for their efforts, they may be unwilling and unable to assume higher risks of lawsuits for breach of warranty.[\[209\]](#) Lowering the risk of lawsuit means "low barriers to entry; anyone can contribute code to the process, not just those who can afford insurance or lawyers to arrange liability-limiting legal structures."[\[210\]](#) Placing the risk of litigation on the open source developer may in turn increase the price of open source products. Another negative consequence is the possible deterrence of programmers from contributing useful code.

{95}Open source advocates are also very concerned about the current UCITA stance on reverse engineering. They claim it causes an "indirect consequence that would hamstring free software development in the long term - it gives proprietary software developers the power to prohibit reverse engineering."[\[211\]](#) Currently, reverse engineering is legal for reasons of "interoperability" between computer systems.[\[212\]](#) Prohibiting reverse engineering inhibits the development of open source because for open source products to be of any value, they must be compatible with other computer applications. The way to establish compatibility is to reverse engineer the other developer's code. Open source advocates are concerned that the UCITA will allow proprietary developers to "establish secret file formats and protocols, which there would be no lawful way for [open source programmers] to figure out."[\[213\]](#) Thus, the UCITA is a double-edged sword. While, it may prove to be a boon, or maybe a bane to open source development. Based on the UCITA language states may choose to adopt, the statute can speed the acceptance of open source software, or, in the alternative, it can bring it to a halt.

{96}To help reduce product concerns, government agencies can use open source products as the software of choice. The government could serve as a test site to show large corporate users that open source software is viable. Several federal agencies have already adopted the Linux operating system.[\[214\]](#) The result of large-scale government involvement in open source could calm many of the marketplace concerns regarding open source products. The Government Accounting Office ("GAO") could commission a study on the reliability and security of open source applications, and the relevant costs associated with a government agency switching platforms.[\[215\]](#) The GAO information would prove to answer corporate concerns about converting to open source.

{97}A government agency could also serve as the standardization or verification entity. If no private entity

steps in to meet this need, the government has the ability to do it. As a government agency, the entity's role would be somewhat different. The role is not to help market open source products, but rather, to be the impartial harbinger of open source information. The agency would provide information to consumers about various versions, the latest updates, patches, and fixes. Additionally, it could employ the expertise of the government and military engineers to guide individuals to the technical solutions sought.[\[216\]](#)

{98}The judicial branch can also embrace the open source ideology. It can be used in antitrust, patent, and copyright violations. Occasionally, the price commercial monopolies pay for antitrust violations is a judicially-enforced corporate dismemberment. Breaking apart a company is difficult, but doing it in the software world offers no real advantage because consumers demand the software interoperability that results from a large company. A more effective remedy is found in allowing the company to remain intact, while denying the company the fruits of its monopoly. Basically, the court can require the monopolistic company to release the source code of its products under an open source license. Similar remedies can be applied to copyright and patent violators. This would allow other developers to enter into the competitive arena without disrupting consumer needs.

{99}AT&T handled the development of UNIX in this way. Federal antitrust laws denied AT&T the right to enter the computer market with UNIX after its development in 1969.[\[217\]](#) AT&T decided the next best idea for UNIX was to give it away to universities. The result was a generation of student programmers who had full access to UNIX source code. GNU systems and the Linux operating system are outgrowths of this action on the part of AT&T. Had AT&T kept UNIX as a proprietary product, open source and its family of inventions might never have come about today.

V. CONCLUSION

{100}This article began with the premise that open source has already created technically valuable products. The products are tested repeatedly by hundreds of developers on hundreds of varying configurations. We have discussed software copyright, licensing, development and distribution methods, and social issues, as they relate to open source. The purpose in discussing these issues is to relieve the concerns of proprietary developers, corporate legal departments, and end users, relating to the relative risks of open source.

{101}Economically, open source is a more efficient way to allocate the benefits of copyright to society. Because current software protection law benefits relatively few developers, there is a need for change. Open source exhibits valid, economical, and marketable alternatives to proprietary software development and distribution.

{102}Regarding copyright infringement, programmers can rest at ease. Derivative works are encouraged under the GPL. Infringement suits are a reality; yet, they are no easier to initiate, nor harder to defend than under a proprietary paradigm. Purchasers are no worse off under GPL warranties than under proprietary end-user warranties. Corporate purchasers who are used to negotiating warranties can still negotiate with open source vendors to provide warranties beyond the GPL.

{103}As vendors move to the open source market, transition issues will arise about how quickly source code is to be made available, how much ownership vendors should retain, and how to appropriately price the software. Open source may renew an interest in computer programming in our society, and can even smooth the information technology employment cycle. Open source has exciting synergistic possibilities with hardware manufacturers. Whether open source will grow to supplant proprietary software methods, or if it just grows to be second best, programmers, purchasers, and users should be open to open source.

^[*] Shawn W. Potter received a B.A. and M.A. (Public Policy) from Brigham Young University in 1996 and he will complete his J.D. at the University of Kansas School of Law in May 2001.

^[**] **NOTE:** All endnote citations in this article follow the conventions appropriate to the edition of THE BLUEBOOK: A UNIFORM SYSTEM OF CITATION that was in effect at the time of publication. When citing to this article, please use the format required by the Seventeenth Edition of THE BLUEBOOK, provided below for your convenience.

Shawn W. Potter, *Opening Up to Open Source*, 6 RICH. J.L. & TECH. 24 (Spring 2000), at <http://www.richmond.edu/jolt/v6i5/article3.html>.

^[1]. See Therese Poletti, *Linux Gains More Corporate Love: HP Allies with Red Hat to Add Linux to Servers, and SGI is Expected to Follow Suit* (visited Feb. 25, 2000) <<http://www.zdnet.com/zdnn/stories/news/0,4586,2195730,00.html>>; see generally *The Linux Home Page at Linux Online* (visited Feb. 25, 2000) <<http://www.linux.org>>; <<http://www.perens.com/OSD.html>> (providing information about Linux).

^[2]. See Steven Brody, *IDC Says Linux Likely to Lead OS Growth: But Getting Exact Numbers for Linux Is Impossible* (visited Feb. 25, 2000) <<http://www.linuxworld.com/linuxworld/lw-1999-03/lw-03-idc.html>>; see also *Other Operating Environments Will Have Trouble Keeping Up with Linux's Growth: Linux Commercial Shipments Will Increase at a Four-Year Rate of 25%* (visited Feb. 25, 2000) <<http://www.idcresearch.com:8080/Data/Software/content/SW033199PR.htm>>.

^[3]. Judy DeMocker, *Behlendorf Jumps to O'Reilly: Move Gives Apache Developer Time to Rearchitect Popular Web Server* (visited Feb. 25, 2000) <<http://www.linuxworld.com/linuxworld/lw-1999-02/lw-02-behlendorf.html>>.

^[4]. See U.S. v. Microsoft: *Findings of Fact* (visited Feb. 25, 2000) <<http://www.callaw.com/stories/findings.html>>; also available at <<http://usvms.gpo.gov/findfact.html>>, and <<http://www.usdoj.gov/atr/cases/f3800/msjudgex.htm>>.

^[5]. See Ira V. Heffan, *Copyleft: Licensing Collaborative Works in the Digital Age*, 49 Stan. L. Rev. 1487, 1492 (1997)[hereinafter Heffan, *Copyleft*].

^[6]. *Id.* at 1492-93.

^[7]. *Id.* at 1494; see also Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. Legal Stud. 321, 327 (1995).

^[8]. See Heffan, *Copyleft*, *supra* note 6, at 1494.

^[9]. Stephen Fishman, *Software Development, A Legal Guide*, 2-3 (2d ed.1998).

^[10]. See Heffan, *Copyleft*, *supra* note 6, at 1493.

[11]. See *id.* at 1496.

[12]. See *The Open Source Definition* (visited Feb. 25, 2000) <<http://www.opensource.org/osd.html>> [hereinafter *Open Source*].

[13]. See generally *Tech. Encyclopedia: Source Code* (visited Feb. 25, 2000) <<http://www.techweb.com/encyclopedia/defineterm?term=SOURCECODE&exact=1>> (defining source code) [hereinafter *Tech. Encyclopedia*].

[14]. See *id.*

[15]. See *Open Source*, *supra* note 13.

[16]. See *GNU General Public License* (visited Feb. 25, 2000) <<http://www.opensource.org/licenses/gpl-license.html>> [hereinafter *GNU GPL*].

[17]. See *id.*

[18]. See *id.*

[19]. See *id.*

[20]. See *Netscape Server Software End User License Agreement* (visited Feb. 25, 2000) <<http://www.netscape.com/download/server.html>>; see generally, *You Got a License for This?* (visited Feb. 25, 2000) <<http://www.computerworld.com/home/online9697.nsf/All/970512license1ink>> (noting the restrictive nature of proprietary software licensing).

[21]. See generally *About Software Licensing* (visited Feb. 25, 2000) <http://www.compaq.com/products/software/info/sw1_about.html> (explaining that proprietary software licenses limit usage to a maximum number of computers).

[22]. *What is Free Software?* (visited Feb. 25, 2000) <<http://www.gnu.org/philosophy/free-sw.html>>.

[23]. See Richard Stallman, *The GNU Project* (visited Feb. 25, 2000) <<http://www.gnu.org/gnu/the-gnu-project.html>> (explaining that GNU stands for "GNU's Not UNIX") [hereinafter Stallman, *GNU Project*]; see also Richard Stallman, *Initial Announcement* (visited Feb. 25, 2000) <<http://www.gnu.org/gnu/initial-announcement.html>> [hereinafter Stallman, *Initial Announcement*].

[24]. Stallman, *Initial Announcement*, *supra* note 24.

[25]. *Id.*

[26]. Free Software Foundation, *The GNU Manifesto*, (visited Feb. 25, 2000) <<http://www.gnu.org/gnu/manifesto.html>> [hereinafter, Free Software Foundation, *GNU Manifesto*].

[27]. Stallman, *GNU Project*, *supra* note 24.

[28]. *Tech. Encyclopedia*, *supra* note 14.

[29]. *Id.*

[30]. Charles Arthur, *The Man Who Can Smash Windows*, *The Independent* (London), July 28, 1999, at 1, 2 [Arthur, *The Man*].

[31]. *Id.*

[32]. *Id.*

[33]. Stallman, *GNU Project*, *supra* note 24.

[34]. *See id.*

[35]. *See id.*

[36]. *Tech. Encyclopedia*, *supra* note 14.

[37]. *See id.*

[38]. *See id.*

[39]. *See id.*

[40]. *Id.*

[41]. *Id.*

[42]. *See id.*

[43]. *See id.*

[44]. *Id.*

[45]. *Id.*

[46]. *See* Tim O'Reilly, *The Open Source Revolution*, Esther Dyson's Monthly Rep., (last modified Nov. 1998) <<http://www.eventure.com/release1/1198.html>> [hereinafter O'Reilly, *Open Source Rev.*].

[47]. *Open-Source Software* (visited Feb. 28, 2000) <<http://www.tux.org/~niemi/opensource/customer-case.htm>>; *see also* Marshall Kirk McKusick, *Twenty Years of Berkley Unix-From AT&T Owned to Freely Distributable* (visited Feb. 25, 2000) <<http://www.oreilly.com/catalog/opensources/book/kirkmck.html>>.

[48]. *See X Consortium* (visited Feb. 28, 2000) <<http://www.x.org/>>; *see also Graphical Interface* (visited February 28, 2000) <<http://www.systemsix.com/prop/xwindows.html>>; O'Reilly, *Open Source Rev.*, *supra* note 47.

[49]. *See Debian* (visited Feb. 25, 2000) <<http://www.debian.org/intro/about>>.

[50]. *Microsoft Museum* (visited Feb. 22, 2000) <<http://www.microsoft.com/Museum/default.asp>>.

[51]. *Id.*

[52]. Mary Jo Foley, *Microsoft Evaluates Open Software 'Threat'*, PCWeek Online (Nov. 2, 1998) <<http://www.zdnet.com/pcweek/stories/news/0,4153,369430,00.html>>; *see also* Stephen Shankland, *Microsoft Spins "Halloween" Memos*, CNET News.com (Nov. 6, 1998) <<http://news.cnet.com/news/0-1003-200-335100.html>>; [hereinafter Shankland, *Microsoft Spins*].

[53]. *See* Shankland, *Microsoft Spins*, *supra* note 53.

[54]. See Arthur, *The Man*, *supra* note 31.

[55]. See *id.*

[56]. See *id.*

[57]. See David Ticoll, *Consider the Open Source*, Tele.com, June 7, 1999.

[58]. Steve Gelsi, *Red Hat Gains 272 Percent IPO Turns in Red Hot Performance*, CBS MarketWatch (last modified Aug. 11, 1999) <http://cbs.marketwatch.com/archive/19990811/news/current/ipo_rep.htm?source=blq/yhoo&dist=yhoo>.

[59]. Roland Moller, *Linux Founder Pleased with Market Hype: Not Jealous of Those Making Millions on His Program*, Financial Post, Dec. 14, 1999, at C12.

[60]. Stephen Shankland, *VA Linux Files IPO Plans*, CNET News.com (last modified Oct. 9, 1999) <<http://yahoo.cnet.com/news/0-1003-200-811842.html>>.

[61]. Glyn Moody, *The Greatest OS That (N)ever Was*, Wired Magazine, Aug. 1997 (quoting Eric Youngdale).

[62]. *Id.*

[63]. See Free Software Foundation, *supra* note 27 and accompanying text.

[64]. See generally OsOpinion, *The Practical manager's Guide to Linus: Can You Profitably Use Linus in Your Organisation?* (visited Mar. 17, 2000) <<http://www.osopinion.com/Opinions/GaneshCPrasad/GaneshCPrasad2.html>> [hereinafter osOpinion, *Practical Manager's Guide*].

[65]. S-1 Prospectus, Red Hat SEC Filing, June 4, 1999.

[66]. *Id.*

[67]. See *History of the Open Source Initiative* (visited Mar. 17, 2000) <<http://www.opensource.org/history.html>>; Charles Cooper & Lisa M. Bowman, *Ballmer: Microsoft Taking Notice of Free Rivals Linux, Apache* (visited Mar. 17, 2000) <http://www.zdnet.com/zdnn/stories/zdnn_smgraph_display/0,3441,2134010,00.html>; *Mozilla News* (visited Mar. 17, 2000) <<http://www.mozilla.org>>.

[68]. See *Open Source Products* (visited Mar. 17, 2000) <<http://opensource.org/products.html>>.

[69]. See Nicholas Petreley, *Corel Controversy* (last modified Oct. 15, 1999) <<http://www.linuxworld.com/linuxworld/lw-1999-09/lw-09-corelbeta.html>>.

[70]. *Id.*

[71]. See Mozilla org., *Mozilla Release FAQ* (visited Mar. 17, 2000) <<http://www.mozilla.org/docs/mozilla-faq.html>> (stating that the open source project was nicknamed "Mozilla") [hereinafter Mozilla.org., *Mozilla Release*].

[72]. N. Drakos, Gartner Group, *An Evaluation Framework for Open-Source Software*, Commentary, June 1,

1999.

[73]. See generally Opinion, *Practical Manager's Guide*, *supra* note 65.

[74]. See Joe Barr, *Eric Raymond Keynote at the Open Source Forum* (visited Mar. 17, 2000) <http://www.linuxworld.com/linuxworld/lw-1999-06/f_lw-06-esr.html> [hereinafter Barr, *Eric Raymond Keynote*]; see also *The Magic Cauldron: Indirect Sale- Value Models* (visited Mar. 17, 2000) <<http://www.tuxedo.org/~esr/writings/magic-cauldron/magic-cauldron-9.html>> [hereinafter Raymond, *Magic Cauldron*].

[75]. See *Magic Cauldron: Indirect Sale-Value Models*, *supra* note 75; see also Barr, *Eric Raymond Keynote*, *supra* note 75.

[76]. *Id.*

[77]. See *id.*

[78]. *Magic Cauldron*, *supra* note 75; see also Eric Raymond, *The Cathedral and the Bazaar* (visited March 26, 2000) <<http://www.tuxedo.org/~esr/writings/cathedral-paper.html>> (discussing Netscape's release of Mozilla) [hereinafter *Cathedral and Bazaar*].

[79]. See *Magic Cauldron*, *supra* note 75; see also OpenSource.org, *The Business Case for Open Source* (visited Mar. 17, 2000) <<http://opensource.org/for-suits.html>>.

[80]. *Magic Cauldron*, *supra* note 75.

[81]. *Id.*

[82]. *Id.*

[83]. *Id.*

[84]. *Id.*

[85]. *Id.*

[86]. *Id.*

[87]. *Id.*

[88]. See Mozilla org., *Mozilla Release*, *supra* note 72.

[89]. See generally Eric Raymond, *Open Source Software A (New?) Development Methodology* (visited Mar. 16, 2000) <<http://www.opensource.org/halloween/halloween1.html>> (presenting an annotated memorandum originating with Microsoft); see also *Techweb* (visited Mar. 16, 2000) <<http://www.techweb.com/news>> (conducting a general search on this site).

[90]. See generally *Open Source Hardware* (visited Mar. 16, 2000) <<http://www.eg3.com/micr/opensrc.htm>> (stating that open source is not just for software anymore).

[91]. See generally *The OpenCPU Source Page* (visited Mar. 16, 2000) <<http://www.opencpu.freemove.co.uk/>> (regarding project to develop an open source CPU that may be used for electronic circuits); Troy Benjergdes, *Industry Analysis Paper* (visited Mar. 16, 2000)

<<http://web.dodds.net/~hozer/opensource.html>> (containing a brief overview of the idea of open-source hardware).

[92]. See Bernard Cole, *Customized RTOS Targets Open-Source Strategy*, *Electronic Engineering Times*, Sept. 28, 1998 [hereinafter Cole, *Customized RTOS*].

[93]. See generally Webopedia, *COBOL* (visited Mar. 16, 2000) <<http://webopedia.internet.com/TERM/C/COBOL.html>> (stating that although disparaged by many programmers for being outdated, COBOL is still the most widely used programming language in the world).

[94]. See *Magic Cauldron*, *supra* note 75.

[95]. See Lawrence Lessig, *The Limits in Open Code: Regulatory Standards and the Future of the Net*, 14 *Berkeley Tech. L.J.* 759 (1999).

[96]. See, e.g., *Microsoft.com Download Center* (visited Mar. 15, 2000) <<http://www.microsoft.com/downloads/search.asp?>> (offering Microsoft's "Internet Explorer" as a free download); *Netscape Products* (visited Mar. 15, 2000) <<http://home.netscape.com/download/index.html>> (offering Netscape's "Navigator" as a free download).

[97]. See *United States v. Microsoft Corp.*, 65 F. Supp.2d 1, 14-15 (D.C. 1999).

[98]. See *Adobe Systems, Inc. v. South Sun Products, Inc.*, 187 F.R.D. 636, 637 (S.D. Cal. 1999).

[99]. See Gary Anthes, *Eyes on the ITWallet*, *COMPUTERWORLD*, Jan. 3, 2000, at 96; see generally *Netron Announces New Capabilities for Finding and Componentizing Business Rules in Legacy Systems*, *Business Wire*, Jan. 25, 2000, available in LEXIS, News Group File (noting that the company is "[a]n acknowledged leader in software reuse technology"). *Id.*

[100]. See, e.g., *Southworth v. Grebe*, 151 F.3d 717, 728 (7th Cir. 1998); *Love v. Reilly*, 924 F.2d 1492, 1495 (9th Cir. 1991).

[101]. See Kenneth W. Dam, *supra* note 8, at 333.

[102]. See 45 Cong. Rec. S8253 (daily ed. July 12, 1999) (comments of Sen Leahy introducing S. 1257, the Digital Theft Deterrence and Copyright Damages Improvement Act of 1999).

[103]. See Oliver Grawe, *Blaring Trial Headlines Ignore Several Key Aspects of Software Industry*, *Legal Times*, Mar. 8, 1999, at S30.

[104]. Michael Risch, *How Can Whelan v. Jaslow and Lotus v. Borland Both Be Right? Reexamining the Economics of Computer Software Reuse*, 17 *J. Marshall J. Computer & Info. L.* 511 (1999) [hereinafter Risch, *Reexamining the Economics*].

[105]. Leander Kahney, *Linux's Forgotten Man* (last modified Mar. 5, 1999) <<http://www.wired.com/news/technology/0,1282,18291,00.html>>. The scientific method is defined as the "principles and procedures for the systematic pursuit of knowledge involving the recognition and formulation of a problem, the collection of data through observation and experiment, and the formulation and testing of hypotheses." Merriam-Webster's Collegiate Dict. 1045 (10th ed. 1999).

[106]. *Magic Cauldron*, *supra* note 75.

- [107]. *Id.*
- [108]. *Id.*
- [109]. See Cole, *Customizing RTOS*, *supra* note 93.
- [110]. See *Redhat.com Store* (visited Feb. 16, 2000) <<https://www.redhat.com/commerce/redhatlinux.html>>.
- [111]. See *id.*
- [112]. See *Redhat.com - How to Download Red Hat Linux 6.1* (visited Feb. 17, 2000) <http://www.redhat.com/download/howto_download.html>.
- [113]. See *id.*
- [114]. *CDW Product Overview* (visited Feb. 16, 2000) <<http://www.cdw.com/shop/products/default.asp?EDC=158754>> (listing the price for Microsoft Windows 98 Second Edition Version).
- [115]. See *id.* (listing the price for Microsoft Office 2000 Professional Upgrade).
- [116]. See ROBERT P. MERGES ET AL., *INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE* 323 (1997) [hereinafter MERGES, *INTELLECTUAL PROPERTY*].
- [117]. See *id.*
- [118]. 17 U.S.C. §§ 101-1332 (1994 & Supp.IV 1998).
- [119]. MERGES, *INTELLECTUAL PROPERTY*, *supra* note 117.
- [120]. See Pub. L. No. 96-517 § 10(a), 94 Stat. 3028 (1980) (amending § 101 of the Copyright Act).
- [121]. U.S. Const. art. I, § 8, cl. 8.
- [122]. 35 U.S.C. § 101 (1994).
- [123]. 17 U.S.C. § 102(a) (1994).
- [124]. See 17 U.S.C. § 102 (1994 & Supp. IV 1998); see, e.g., U.S. Copyright Office, *Copyright Basics* (last modified Sept. 27, 1999) <<http://lcweb.loc.gov/copyright/circs/circ1.html>> (giving a primer on the Copyright Act).
- [125]. See 17 U.S.C. § 106 (1994 & Supp. IV 1998).
- [126]. See U.S. Const. art. I, § 8, cl. 8.
- [127]. See Risch, *Reexamining the Economics*, *supra* note 105.
- [128]. See *id.*
- [129]. *Id.* (emphasis added).
- [130]. 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, 185-86 (1999)

[Gomulkiewicz, *Copyleft Uses License Rights*]; see, e.g., *Open BSD Copyright Policy* (last modified Sept. 29, 1999) <<http://www.openbsd.org/policy.html>> (noting that while OpenBSD is copyrighted, it is freely redistributable).

[131]. See *Open Source*, *supra* note 13.

[132]. 17 U.S.C. § 302(a) (Supp. IV 1998).

[133]. See *Microsoft Timelines* (visited Feb. 28, 2000) <<http://www.microsoft.com/Museum/musTimeline.asp>>.

[134]. See *id.*

[135]. See *id.* (noting that in the time line of Microsoft product releases Windows was upgraded in 1987, 1990, 1992, and 1995, with the Workgroup and NT versions released in 1993). While not yet listed in the *Timeline*, the most recent upgrades of Windows were released in 1998 and 2000.

[136]. See Daniel J. Langin, *Insurance Coverage for IP Infringement -- Helping Companies Protect Themselves*, 3 CYBER. LAW. 8 (1998).

[137]. See *GNU GPL*, *supra* note 17.

[138]. See *id.*

[139]. *Id.*

[140]. *Id.*

[141]. *Id.*

[142]. *Id.*

[143]. *Id.*

[144]. *Open Source*, *supra* note 13.

[145]. See *GNU Lesser General Public License* (last modified Feb. 1999) <<http://www.opensource.org/licenses/lgpl-license.html>>.

[146]. E-mail from K. Powers, Staff member of Cygnus Solutions, to Shawn Potter (June 29, 1999) (e-mail on file with the author).

[147]. *Id.*

[148]. See Heffan, *supra* note 6, at 1492-93. For example, a search conducted on March 26, 2000 in the LEXIS Copyright Law, Federal Cases Database for "open-source" or "open source" did not produce any results in terms of case citations.

[149]. See *id.* at 1509-10.

[150]. E-mail from K. Powers, Staff member of Cygnus Solutions, to Shawn Potter (June 29, 1999) (e-mail on file with the author), *supra* note 147.

[151]. See 17 U.S.C. § 501 (1994).

[152]. See 17 U.S.C. § 201 (1994).

[153]. H. R. REP. NO. 94-1476 (1976), *reprinted as* HISTORICAL AND REVISION NOTES to 17 U.S.C. § 501 (1994).

[154]. *Broadcast Music, Inc. v. CBS, Inc.*, 221 U.S.P.Q. 246 (S.D.N.Y. 1983) (citing *Bertolino v. Italian Line*, 414 F. Supp. 279, 284 (S.D.N.Y. 1976)).

[155]. See *GNU GPL*, *supra* note 17.

[156]. E-mail from Brian Youmans, Free Software Foundation, to Shawn Potter (Oct. 28, 1999) (e-mail on file with the author).

[157]. See generally Heffan, *supra* note 6, at 1496-96 (discussing copyright protection of software).

[158]. 17 U.S.C. § 501(b) (1994).

[159]. *Sun Microsystems, Inc. v. Microsoft Corp.*, 188 F.3d 1115, 1121 (9th Cir. 1999) (quoting *Graham v. James*, 144 F.3d 229, 236 (2nd Cir. 1998)).

[160]. *Id.*

[161]. See *infra* notes 163-166 and accompanying text (discussing cases holding that GPL breaches equate to copyright infringement).

[162]. See, e.g., *Tingley Sys., Inc. v. Norse Sys., Inc.*, 49 F.3d 93, 98 (2d Cir. 1995) (upholding jury finding of no infringement where jury question stated that "unless Tingley proved an agreement . . . that Tingley software could only be used on the Ultimate machines, Tingley could not prevail on its copyright claim"). *Id.*; see also Stephen J. Sand, *Validity, Construction, and Application of Computer Software Licensing Agreements*, 38 A.L.R. 5th 1, 10 (1999) (citing cases in which courts found that a breach of a software license resulting in a use that exceeds the scope of the license constitutes copyright infringement).

[163]. See *Advanced Computer Servs. v. MAI Sys. Corp.*, 845 F. Supp. 356 (E.D. Va. 1994).

[164]. See *SAS Inst., Inc. v. S & H Computer Sys.*, 605 F. Supp. 816, 828 (M.D. Tenn. 1985).

[165]. See *id.* at 827.

[166]. See *GNU GPL*, *supra* note 17.

[167]. See generally Michael Liberman, Comment, *Overreaching Provisions in Software License Agreements*, 1 Rich. J.L. & Tech. 4, ¶ 39 (Apr. 10, 1995) <<http://www.richmond.edu/jolt/v1i1/liberman.html>> (noting that in *Lasercomb Am., Inc. v. Reynolds*, 911 F.2d 970 (4th Cir. 1990), the Fourth Circuit found the non-compete clauses in defendant's license agreement were contrary to public policy).

[168]. See *Lasercomb Am., Inc. v. Reynolds*, 911 F.2d 970, 978-79 (4th Cir. 1990).

[169]. *Tamburo v. Calvin*, No. 94 C 5206, 1995 WL 121539, at *7 (N.D. Ill. Mar. 17, 1995).

[170]. See generally *The Business Case for Open Source* (visited Mar. 15, 2000) <<http://www.opensource.org/for-suits.html>> (describing the various products that have been created).

[171]. See Pub. L. 106-160, 113 Stat. 1774 (1999), *amending* 17 U.S.C. § 504(c)(2) (Supp. IV. 1998).

[172]. See 17 U.S.C. § 502(a) (1994).

[173]. See *Cadence Design Sys., Inc. v. Avant! Corp.*, No. 99-15048, 1999 WL 561261, at *1 (9th Cir. July 30, 1999).

[174]. See *Open Source Definition*, *supra* note 13.

[175]. *Id.*

[176]. See *GNU GPL*, *supra* note 17.

[177]. See *id.*; Microsoft Corporation, *Microsoft End User License Agreement (EULA): Windows 98 - Online (version 3/26/98)* (visited Feb. 25, 2000) <<http://www.linuxmall.com/misc/refund/eula/online032698.html>>. (reproducing the Microsoft End User License Agreement in whole) [hereinafter Microsoft, *License Agreement*].

[178]. See *GNU GPL*, *supra* note 17; Microsoft, *License Agreement*, *supra* note 178.

[179]. See *Microsoft Windows 98 End User license Agreement, OEM Version 4/10/1995* (noting that a copy of the End-User License Agreement is included in every Windows 98 software packet); Microsoft, *License Agreement*, *supra* note 178.

[180]. See *GNU GPL* *supra* note 17.

[181]. See Microsoft, *License Agreement*, *supra* note 178.

[182]. See *GNU GPL*, *supra* note 17.

[183]. Microsoft, *License Agreement*, *supra* note 178.

[184]. *Id.*

[185]. *GNU GPL*, *supra* note 17.

[186]. Microsoft, *License Agreement*, *supra* note 178.

[187]. See *GNU GPL*, *supra* note 17.

[188]. See Microsoft, *License Agreement*, *supra* note 178.

[189]. See *GNU GPL*, *supra* note 17.

[190]. See Microsoft, *License Agreement*, *supra* note 178; *GNU GPL*, *supra* note 17.

[191]. See E-mail from Craig Ozancin, Engineer for Axent Technologies, to Shawn Potter (Oct. 6, 1999) (e-mail on file with the author).

[192]. See *id.*

[193]. See *id.*

[194]. *See id.*

[195]. *See id.*

[196]. E-mail from Steve Jackson, Engineer for Axent Technologies, to Shawn Potter (Oct. 6, 1999) (e-mail on file with the author).

[197]. *See* Paul Ferris, *Fixing Security Holes on Internet Time*, LINUX TODAY (last modified June 21, 1999) <<http://linuxtoday.com/stories/6947.html/>> (noting the users fixed a bug in a matter of hours)[hereinafter Ferris, *Fixing Security*] ; *see also* E-mail from Craig Ozancin, *supra* note 192 (stating that problems may be fixed relatively quickly because the source code is widely accessible).

[198]. *See generally* E-mail from Craig Ozancin, *supra* note 192 (alluding to the fact that proprietary software from producers like Microsoft has a longer wait period for locating and fixing problems, based on the fact that the source code is not accessible to everyone).

[199]. Ferris, *Fixing Security*, *supra* note 198.

[200]. *Id.*

[201]. Free Software Foundation, Inc., *Frequently Asked Questions About Open Source: Doesn't Closed Source Help Protect Against Crack Attacks?* (visited Feb. 24, 2000) <<http://www.opensource.org/faq.html>>.

[202]. *See* Ferris, *Fixing Security*, *supra* note 198.

[203]. *See* Email from Craig Ozancin, *supra* note 192.

[204]. *See id.*

[205]. *See generally* GNU GPL, *supra* note 17 (allowing users to modify the program).

[206]. Information regarding Underwriters Laboratories ("UL") services can be found at Underwriters Laboratories, Inc., *About UL -- Who We Are and What We Do* (visited Mar. 26, 2000) <<http://www.ul.com/about/index.html>>.

[207]. *See* ProCD, Inc. v. Zeidenberg, 86 F.3d 1447 (7th Cir. 1996) (holding that a shrinkwrap license to be binding on a software buyer under the UCC).

[208]. *See* Gomulkiewicz, *Copyleft Uses License Rights*, *supra* note 131, at 190.

[209]. *See id.* at 191-92.

[210]. *Id.* at 192.

[211]. Richard Stallman, *Updated: Richard Stallman -Why We Must Fight UCITA*, LINUX TODAY (last modified Feb. 6, 2000) <<http://linuxtoday.com/stories/15948.html>> [hereinafter Stallman, *Updated*].

[212]. *See* 17 U.S.C. § 1201(f)(1) (Supp. IV. 1998).

[213]. Stallman, *Updated*, *supra* note 212.

[214]. *See, e.g.*, Dana Gardner, *Linux 2.2 Boosts Server Scaling*, INFOWORLD (last modified Nov. 16, 1998) <<http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/46/t06-46.8.htm>> (noting use by the U.S. Navy's

Fleet Numerical Meteorologic and Oceanographic Center, in Monterey, California); Mitch Stoltz, *The Case for Government Promotion of Open Source Software* (visited Jan. 29, 2000) <<http://www.netaction.org/opensrc/oss-recommend.html>> (noting the U.S. Postal Service's use of a modified version of Linux) [Stoltz, *Government Promotion*]; see generally Torsten Busse, et al., *Linux Continues to Pick up Steam: Customers, Vendors Wrestle with Issues But Remain Excited*, INFOWORLD (last modified Nov. 30, 1998) <<http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/48/c08-48.34.htm>> (reporting recent corporate interest in Linux).

[215]. See Stoltz, *Government Promotion*, *supra* note 215 (stating that a Congressional initiated study by the GAO would be a "risk-free way to assess the benefits of OSS [open source software] to particular government agencies"). *Id.*

[216]. See *id.*

[217]. See *Unix for the Masses*, LAN TIMES, August 1990, at 44.

Related Browsing

[Copyright 2000 Richmond Journal of Law & Technology](#)