Canadian Journal of Law and Technology

Volume 1 | Number 3 Article 3

8-1-2002

Open Source, Open Arms: An Open-Ended Question

Alana Maurushat

Open Source, Open Arms: An Open-Ended Question

Alana Maurushat†

Introduction

The proliferation of computer technology and the advent of the Internet raise novel questions about traditional legal, economic and philosophical principles. Rethinking how software technology is held, developed and distributed is a growing movement in technology. Led by a group of "hackers", the movement has come to be known as "open source" although it is often associated with "freeware" and "copyleft". Each term generically describes the movement although these terms imply different ideas to those inside of the open source movement.[1] Open source represents a community; a community comprised of computer programmers, distributors and users, each, in varying degrees, committed to a common goal — that source code should be freely available while users should be able to modify source code without violating the software licensing agreement.

The open source software movement poses a profound challenge to the way that software is made and distributed. Projects are established and programmers will communicate and contribute software building blocks to one another via the Internet. When a software program is completed by this method, it is then generally offered to the public over the Internet, sometimes free of charge, but always free of the use restrictions common to most software. In this respect, open source software differs from most proprietary software in two ways: first, the holder of a copy of some open source software is at liberty to make unlimited copies, to modify the code, and to further distribute copies; and second, open source software is distributed with access to the source code, not just the object code.[2] These conditions are promulgated through the innovative use of software licensing.

This paper is structured to address several aspects and challenges to the open source movement. Beginning with an outline of the historical and cultural components of the open source movement, the paper will move on to explore the economic and philosophical underpinnings of intellectual property. It will be demonstrated that open source finds itself uniquely situated within these theories and doctrines. The questions that open source poses for intellectual property will then be examined. My arguments will stem from the general premise that open source is threatened by three mechanisms: the uncertainty of the validity of open source licenses, potentially over-expansive copyright law, and by the growth in computer software patents. The core of this paper will outline the beneficial aspects of open source: it provides competition to those few corporations who currently dominate the software market; it presents a viable alternative to traditional economic models for software; and it protects fundamental societal values such as free speech by acting as a counter force against governmental and market control of code. The conclusion crystallizes the underlying notion of this paper: should intellectual property laws be amended or interpreted in order to foster open source? Given the important and invaluable economic and social role open source plays in the computer software industry, legislative and regulatory measures should be developed to promote and encourage open source.

What is Open Source?

History and Emergence of the Open Source Movement

The principles of free modification and free distribution of source code were institutionalized in 1985 by Richard Stallman, who founded the Free Software Foundation to encourage software development based on these principles.[3] Developers who subscribed to the principles of free distribution and modification of software became known as the "free software" community.[4] Use of the word "free" in this context connotes non-proprietary, not necessarily non-commercial. As Mr. Stallman puts it, "Think 'free speech', not 'free beer'".[5] Richard Stallman wrote a complete UNIX-compatible software system that he named GNU when he became dissatisfied with the way developers restrictively licensed software.[6] Most consider this to be the beginning of the open source movement. Stallman, a former computer programmer and researcher at the MIT Artificial Intelligence Lab, chose UNIX because it was portable, flexible, and a powerful multi-tasking operating system.[7] He wanted to, "give (his GNU system) away free to everyone

†Lecturer, Faculty of Law, University of Hong Kong. LLM With Concentration in Law and Technology, University of Ottawa. BCL and LLB, McGill University.

who [could] use it".[8] Stallman asked manufacturers for donations of machines and money and individuals for donations of programs and work. As more programmers became involved, Stallman issued the GNU Manifesto to 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.[9]

GNU gradually gained contributors, mainly in the form of academics and hobbyist programmers. Progress on the GNU project proceeded slower than the proprietary software environment, but by the 1990s all major components except one, the kernel, had been found or written. The kernel is the part of the operating system that activates the hardware directly or interfaces to another software layer that drives the hardware.[10] It is a fundamental part of the program. This problem was solved by Linus Torvald's development of such a kernel, now known to us as Linux. With the contribution of Linux, GNU acquired a fully-functioning, UNIX-compatible operating system.[11]

As GNU's popularity grew, primarily among programmers and academics, Stallman and others founded the Free Software Foundation ("FSF"), an organization designed to promote free software development.[12] The FSF assumed control of the distribution of GNU and free software that operated on the GNU system.[13]

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 term was to avoid the confusion of the word "free" with "gratis".[14] Other community members, however, were motivated to use "open source" as an alternative method of programming and business structure. 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 they represent different ideas and values about the software.[15]

Although the free software community zealously believed in the superiority of its approach to software development, in the beginning, free software products barely made a ripple in the marketplace.[16] The most successful free software products were tools for software developers. Hackers used software created by other hackers, but business and consumers continued to use commercially developed software products. The Internet changed that equation.

Many of the software programs integral to the infrastructure of the Internet and World Wide Web are open source software programs.[17] The software program known as BIND allows website addresses to be written in plain English.[18] The Sendmail electronic mail router directs virtually every piece of email sent over the Internet. The Apache web server is the most popular web server software for hosting web sites. Furthermore, free software languages such as Perl, Java, TcL, and Python are used in the development of popular websites such as Yahoo! and Amazon.com.[19]

This quiet revolution became a public event in January 1998 when Netscape shocked most people by announcing that it would give away the source code to its Navigator web browser software.[20] Netscape's move was inspired, at least in part, by a paper, which was later expanded to a book, written by hacker Eric S. Raymond, entitled, "The Cathedral and the Bazaar".[21] Raymond argues that software development based upon an open source model is technically superior to software developed by teams employed by commercial software developers.[22] At about the same time, a free software product known as the Linux operating system began to grow in popularity. Linux became known as the operating system product that would challenge popular products such as Windows, Windows NT, and various UNIX derivatives such as Solaris and SCO UNIX. As many have suggested, it has been the overwhelming and rapid growth of the Linux operating system that has been the vehicle behind the open source movement.[23]

Soon the press was writing about the open source software movement, and commercial software publishers were taking actions in response.[24] For example, IBM included the Apache Group's web server in its WebSphere server suite. Oracle announced that it would port its database to Linux. Intel made an investment in Linux distributor Red Hat Software. Corel said it would release a free version of its WordPerfect word processing product for the Linux platform. Sony announced that PlayStation 2 would run on a Linux platform. According to Robert Gomulkiewicz, "the open source movement went from a footnote to an exclamation point; from obscurity to a force to be reckoned with".[25]

Factions of Ideology

Within any movement there exist factions of the group whose ideologies will vary. The open source movement is no exception. One author has even gone so far as to hypothesize that these differing ideologies will fragment the movement leading to its demise.[26] The belief in the motto, "united we stand, divided we fall", may be exaggerated within the open source movement. Although there are varying ideologies within the open source community, the common goal of developing a new method of software creation and distribution provides the cohesion necessary for the survival of the movement.

At one end of the open source community is a faction of the movement led by Richard Stallman, who has been fighting against proprietary interests in com-

puter programs for the past 15 years through the Free Software Foundation now known as the "copyleft" movement. The FSF believes that protection of software through current intellectual property regimes impedes innovation, contributes to monopolies and is ethically wrong.[27]

Stallman and many members of FSF oppose the application of intellectual property law to software.[28] Recognizing ownership in software, as Stallman explains, has detrimental material effects on society: it makes programs more expensive to construct and distribute, less efficient to use, and obstructs use by not allowing users to adapt or fix programs.[29] In addition to material harms, Stallman argues that there are "psychosocial harms" attributable to the proprietary software model which degrade relationships among fellow citizens. As Stallman philosophizes:

> My work on free software is motivated by an idealistic goal: spreading freedom and cooperation. I want to encourage free software to spread, replacing proprietary software that forbids cooperation, and thus make our society better.[30]

Stallman's moralistic beliefs about the abandonment of copyright have reached an almost religious level, and have inspired a "cult-like" following for the FSF movement. Stallman has been called "impracticably messianic" and "a fanatic with an unrealistic, uncompromising vision" while in the same breath being cited as someone with a vision that "you can't ignore".[31] Some view Stallman's radically socialist ideology and his ethical and moralistic stance against proprietary software as a form of "zealotry".[32] Stallman urges his followers to recognize the moral and social importance of free software, "free software is a matter of liberty, not price".[33] Stallman's uncompromising stance on the underlying ethical issues propelling his advocacy of free software helps frame his criticisms of intellectual property in general. Worried about measures taken to protect proprietary interests in software, Stallman criticizes the "increasingly nasty and draconian measures now used to enforce software copyright".[34] He notes that the motive for information control and intellectual property in the United States is profit. He strongly believes that current intellectual property regimes are a threat to free speech stating, "when the traditional methods of protecting … ownership have become ineffectual, attempting to fix the problem with broader and more vigorous enforcement will inevitably threaten freedom of speech".[35]

For many years, the Free Software Foundation led by Stallman, was the only sponsor of open source with an institutional identity visible to outside observers of the hacker culture. They effectively defined the term "free software" deliberately giving it a confrontational weight.[36] Thus, perceptions of the hacker culture tended to identify the culture with the FSF's zealous attitude and perceived anti-commercial aims.

On the other side were the quieter, less confrontational and more market-friendly strain in the hacker culture who took a less extreme stance. It was this group of people, the "pragmatists", who gave the movement the new label of "open source" to deliberately avoid giving the movement confrontational weight. The typical pragmatist attitude is only moderately anti-commercial, and its major grievance against the corporate world is not dominating control over property, but the world's perverse refusal to adopt superior approaches incorporating the Unix-type model of open standards and open source software.[37] That is, pragmatists view open source as a means rather than an end in itself. It is a tool for encouraging software sharing and the growth of bazaar-mode development communities.

The pragmatists found their power base with the explosion of Linux in late 1993.[38] Linus Torvald's theory of free software did not condone the commercial growth of the Linux industry, and he endorsed the use of commercial software for certain specific tasks. Torvald's less fanatical view of the movement began to attract his own group of supporters. Additionally, the rapid growth of Linux attracted many new hackers into the community. Linux, however, was their principle loyalty while the FSF's agenda became more of a side-interest.[39] The pragmatists defend open source software on the grounds of its superior method of development and not primarily on its moralistic principles.

Open Source as a Culture

The open source movement is more than a user–developer model of software development; it has evolved into a culture with its own customs, traditions and expectations. It has been described as a "community of people as a locus of innovation, where knowledge, practice, and technological artefacts are interdependent parts of an evolving social system … This development model is a heterogeneous network of communities and technologies".[40] Others have characterized the open source community as a "gift culture".[41] In gift cultures, social status is determined not by what you control but by what you give away.

Key to gift cultures is the notion of reputation.[42] There are several reasons why community reputation may lead to active participation. It has been suggested that the strongest incentive is the pleasure of a good reputation itself.[43] Prestige within the open source community may allow a programmer to more readily persuade others to join projects developed by such a person, or to place particular value on that person's input.[44] The prestige of a developer in the open source community may additionally place them in higher demand within that market. While the gift-culture idea may be counter-intuitive to many businesses, it has been used to explain philanthropy and donation of resources in general.[45]

Another indication of why users may become developers is to maintain control over the source code. Thus, many of the users that are drawn to open source software are drawn by the prospect of being able to

make their own changes to the code.[46] The interaction between project developers and users may facilitate turning users into co-developers. Examples from open source projects indicate that a combination of encouraging participation among users, specifically soliciting comments regarding design decisions, implementing suggested changes and praising users when they provide patches and feedback, leads to further participation.[47] This helps encourage users, who may already be inclined to make improvements to the code, to resubmit these improvements to the project.

Egotistical behaviour and esteem-seeking is generally not approved of within the open source community. However, esteem-seeking behaviour may actually help drive participants to a higher standard of contribution.[48] The norm helps assure that "one's work is one's statement".[49] This, in turn, ensures that the participants are driven toward a high level of performance, because rewards only come from a peer determination of program quality. Thus open source ownership customs provide a background which esteem may be granted or withheld in the community. Further, because code from self-promoting individuals is not rewarded, such "noise" is filtered out of the open source development discourse. Finally, self-aggrandizement is inconsistent with the quality of intelligent selection of code, necessary for a good project leader.[50] It is also inconsistent with the proper distribution of esteem by the leader to contributors, necessary for sustained user–developer contributions.

Economic and Philosophical Underpinnings

Open source can be seen as inconsistent with traditional economic and philosophical approaches. Upon closer examination, however, open source finds itself uniquely situated within the varying theories: the incentive theory,[51] the property theory,[52] and the Spartan theory.[53]

It has generally been thought that an economic reward is necessary to spur creation of works. This, however, does not explain why thousands of developers writing software are freely giving their software away under the open source license. Thus, the incentive theory of copyright seems to be inapplicable. Likewise, success of the free distribution and alteration of open source software seems to refute the property approach. Rather than the creator of software reaping the rewards, it is freely distributed.

At first glance, it would appear that open source does not lend itself well to the incentive theory of intellectual property. Many open source authors create code by incentives other than the economic rewards so readily associated with copyright. Such non-economic incentives would include the love of elegant problem solving, status among their peers, the wish to further computer science and make things better generally, and even animosity toward commercial software developers.[54] Under traditional notions of economic analysis of law, it was thought that people were predominantly motivated by monetary considerations. Open source provides a particularly striking example of an incentive model based on non-monetary considerations. Indeed, open source programmers are motivated by a diverse range of incentives. When one considers the return in terms of increased status among software development peers, ample incentives become apparent.

The incentives for open source software are aligned with personality theory. This theoretical basis for intellectual property protection is the idea that creative works and inventions embody the personality of the artist or inventor, and accordingly should be protected from physical and intangible harms.[55] In this view, legal protection of intellectual creations is necessary to permit individuals to achieve self-actualization.[56] As previously discussed, open source developers will produce code without the conventional economic incentives provided by copyright protection. Open source licenses freely give up almost all exclusive rights such as free copying, distribution, and modification. However, open source developers do not give up the protection of their reputation; they strictly require that the original author receive credit for his or her contribution.

The protection of reputation — that the work be properly attributed to the author and that any changes in the code are not misattributed to the author — is derived from the European approach to intellectual property.[57] This notion is conceptualized as moral rights or *droit d'auteur*. It is at the core of copyright in many European nations as well as in Canada.[58] The open source movement demonstrates that creators of computer software are no less interested in their status as authors and the integrity of their works than are traditional creators such as songwriters or sculptors. For these reasons, open source is reconcilable with incentive theory.

The open source movement might also be used as an example in favour of the property approach, that is, the theory that intellectual property rights should be as expansive as practicable.[59] The underlying idea behind the property approach is grounded in an efficiency rationale: intellectual property laws serve to promote the more effective use of information, by giving individuals the incentive to exploit it, rather than letting free access by all lead to waste.[60] Privatization of property is believed to be necessary to prevent waste.

The property theory parallels many notions espoused by John Locke. Locke would consider intellectual property appropriate where, first, the production of ideas requires a person's labour; second, that these ideas are appropriated from a "common" which is not significantly devalued by the idea's removal; and third, that ideas can be made property without breaching the non-

waste condition.[61] Open source software fits strangely into that theoretical framework.

Consider the first condition. Writing software requires a person's intellectual labour; software requires considerable work to design, implement, debug and revise. In this process, the third condition of non-waste is also met. Recall that the open source movement relies on keeping software under copyright and distributing it under a license, rather than putting the software into the public domain where anyone can do with it what they will. The open source licenses impose two key restrictions: (1) the licensee may not restrict distribution of the code, and (2) the licensee must make the source code available to others. In essence, the openness of the source code is a strong deterrent to leave the code under-utilized.

The second condition, however, potentially poses a problem. Under this central idea to Locke's justification of property, property can only be appropriated if the net effect does not diminish the commons.[62] Thus, a tract of land can be put into the private hands of a farmer because he will then have an incentive to use it productively and sell his harvest to the public. According to Locke, such use is more productive than leaving the land to lie fallow. Traditionally, the proprietary or "closed" model of software development seems to best comply with this Lockean condition, but the whole point of an open source license is to leave the code open for use by others. Thus, rather than diminishing the commons, the copyright in open source protects and may even expand the commons.[63]

The Spartan approach most clearly resembles the spirit of the open source movement. The Spartan theory is a reaction to the recent expansion of intellectual property law.[64] Under this approach, intellectual property rights are seen as a necessary evil. Copyright restricts freedom of expression; patent restricts research and the utilization of technology; and trademark restricts competition.[65] All three types of restrictions, however, have countervailing benefits such as the promotion of a stable and constant system. Proponents of the theory argue that, due to the high cost such restrictions impose, intellectual property laws should be sharply tailored.[66] For example, copyright should only apply to works to the extent that copyright protection is necessary and desirable to promote the creation of such works.[67] Patent regimes should require source code to be placed in the public domain on the expiry of the patent. Trademark protection should be less focused on the broad protection of brand names and concentrate on consumer deception.[68]

The open source movement is based on the idea that software should be freely distributed and revised. Restricting access to source code limits innovation and promotes the inefficient development of software. In the eyes of some, restricting source code is morally wrong.[69] So, if there were to be intellectual property in software, one might think it should be as minimal as possible. However, as discussed above, open source developers rely on intellectual property laws to prevent certain uses of open source software through licensing regimes.[70]

Intellectual Property and Open Source

The legal structure of open source is an elegant and robust use of intellectual property law.[71] This abandonment of the customary uses of intellectual property law, which are normally used to guard exclusive rights, to safeguard free access to and use of software, demonstrates the unique and creative aspect of open source development.

Copyright and Open Source Licensing

Canadian copyright law is both "author-centered"[72] and "rights-centered". The latter operates by granting rights to copyright holders to promote innovation and to provide economic reward for this innovation.[73] Digital technologies and telecommunications have created an enormous change in the way that information is distributed. Amendments have been made accordingly to cover new forms of works protected by copyright. Prior to the 1988 amendments to the Canadian *Copyright Act*,[74] there was no express reference to computer programs. The definition of "computer program" in the Act followed the principles established in case law prior to this modification whereby it was generally accepted that a computer program, in its written and source code version, was included within the definition of a compilation.[75] Computer software programs became covered as "literary works" in the 1988 amendments.[76]

The amendments to the Act also contain specific exceptions from infringement relating to computer programs. It is not copyright infringement of a computer program for a person who owns a copy of the computer program to make a single copy for personal use or for back-up purposes, or to alter the program for the purpose of compatibility with another computer or operating system.[77]

Copyright infringement is complicated in a number of ways with respect to computer programs. First, object code is a form which is not readily comprehensible, therefore, it is difficult in that form to determine if copyright infringement has occurred.[78] Second, it is often difficult to separate ideas which are unprotectable from protectable expression.[79] Third, outside factors such as the hardware of the program may limit ways in which programmers can create computer programs. Lastly, a significant part of many programs may consist of common programming techniques and language which are reproduced in a multitude of programs and are part of the public domain.[80]

Application of the basic principles of copyright to computer software have proven problematic to courts for the above-mentioned reasons. The inconsistency of Canadian courts coupled with the ambiguity as to what is the appropriate test to apply in Canada, leaves doubt about the scope of copyright protection of computer programs.[81] There is further uncertainty as to what programmers can legitimately borrow or copy from existing programs.

Under an open source regime, some of this uncertainty is diminished. Programmers can borrow and copy as much or as little as they would like providing they comply with the terms of the license. This has typically meant that the source code of the software be kept open, therefore, preventing it from becoming proprietary, and that the author(s) of the code be acknowledged.

To stay open, software must be copyrighted and licensed.[82] An essential component to open source programming is licensing. Because computer programmers are able to obtain copyright in their software programs, the rights derived from copyright enable them to mass license the product. This liberal licensing arrangement allows open source software to continue to distribute software on its own unique terms.

The key to the success of open source is the licensing regime of open source software and development projects. As one open source member put it, "open source lives and dies on copyright law".[83] The proponents of open source software rely on owning the copyright in the code and then licensing it according to a very particular mass-market licensing model.[84] Software is licensed, rather than placed in the public domain, in order to control what is done with the code. Licensing allows code authors to perpetuate their particular software development and distribution model. Without licensing, the open source software development model would be nothing more than an honour system.

Most software publishers choose licensing as a transaction model for the same reasons. The distinction between open source software and "closed" or proprietary commercial software is not one based on the absence of a license in one case and the presence of a license in the other case, but is predicated on the absence or presence of certain license terms.

Proprietary software licenses are often characterized by the restrictions they impose on the users of the software. Licences usually restrict software to "execute-only format" and limit the number of installations allowed per copy of the software.[85] The source code is rarely made available or if it has been made available, it is for limited purposes.

The licensing terms of software are critical to the determination of whether it meets the formal open source definition. There are a number of different types of licenses that are consistent with the requirements of the open source definition. The most common and widely-used of these licenses is the General Public License ("GPL"). The GPL is issued by various open source software distributors in the form of a mass license. 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. The basic requirements of the GPL are that "enhancements, derivatives, and even code that incorporates GPL's code are also themselves released as source code under the GPL".[86] Thus, modifications to GPL software cannot be made closed-source, and no GPL program can be incorporated into a proprietary program. In contrast to a GPL, a proprietary software license restricts licensees to use-only, and changes or enhancements are prohibited.

Other open source licenses include, but are not limited to, the GNU Library GPL (LGPL), the BSD-Style License, the Mozilla Public License, the Netscape Public License (NPL), the Berkeley Software Distribution License, the Aladdin License and the Artistic License.[87] The various existing open source licenses all differ in some details. Open source licensing is, however, based on several key principles. These principles are embodied in The Open Source Definition, published by the Open Source Initiative, and in sample licenses published by the FSF.[88] If a license does not comply with these principles, the software cannot be labelled "open source".[89] The general principles are as follows[90]:

1. Unencumbered Redistribution[91]
2. Source Code Form[92]
3. Derivative Works[93]
4. The Author's Attribution and Integrity[94]
5. No Warranties[95]
6. Self-Perpetuating License Terms[96]
7. Non-Discriminatory[97]
8. Non-Contamination[98]

It is generally acknowledged that the use of mass-market licenses is crucial to software publishers; the open source movement could not operate without non-negotiated, standard-form, "take-it-or-leave-it" mass-market licenses. The open source license transaction takes place between two anonymous parties over the Internet based on the licensor's standard form. The licensee typically manifests assent by clicking an "I agree" button or by using, modifying, or distributing the software. The license terms are non-negotiable because the open source licensing model depends upon certain license terms being in the license agreement.[99] Without those terms, the software being licensed cannot be considered open source software. Moreover, the open source licensing model demands that the licensee sub-license using those exact terms to other licensees of the software.

The enforceability of the GNU or GPL license has not yet been litigated either in the United States or in Canada.[100] The lack of litigation may reflect the values in the open source community, namely that of cooperation

and compromise. For example, the open source development team working on the XVID project has publicly announced that it will cease its work until Sigma Designs complies with the GNU license.[101] XVID alleges that Sigma Design's REALmagic MPEG-4 Video Codec software incorporates a significant amount of the XVID source code. Sigma Design, on the other hand, has claimed ownership of this code and contends that they have neither infringed copyright nor the license terms of the GNU. The XVID team has chosen a non-litigious means of resolving this dispute; they have chosen to cease work on the project and to publicly denounce Sigma Design. This conciliatory attitude may partially explain the lack of litigation surrounding GNU and GPL licenses.

GNU and GPL licenses are similar to shrinkwrap and clickwrap licenses in that they contain unconventional contract formation procedures.[102] As these licenses borrow conceptually from shrinkwrap and clickwrap licenses, it has been suggested that courts will likely look to caselaw on such licenses to determine whether an open source license in enforceable.[103] To the extent that a Canadian court would favour an analogy to a clickwrap license over a shrinkwrap license is merely one of speculation. However, the validity of GPL licenses is thought by many academics to be analogous to shrinkwrap licensing.[104] The enforceability, as will be demonstrated, of open source licenses remains largely ambiguous.

Clickwrap licenses are enforceable in Canada.[105] The validity of clickwrap agreements is further reinforced by Canadian electronic commerce legislation which states that: "The acceptance of an offer may be expressed by an action in electronic form, including touching or clicking on an appropriately designated icon or place on a computer screen. [And that] a contract shall not be denied legal effect or enforceability solely by reason that an electronic document was used in its formation."[106] Nevertheless, parties relying on this format still face the requirement of providing reasonable notice to the signing party. If a court, therefore, analogized an open source license to that of a clickwrap license, the license would likely be enforceable.

Ambiguity, however, surrounds the validity of shrinkwrap licenses on the Canadian front. The issue has yet to be litigated. Should the validity of shrinkwrap licenses be litigated in Canada, it is possible that a Canadian court would look to the United States jurisprudence for guidance. In the American case of *ProCD v. Zeidenberg*,[107] the court held that standard form shrinkwrap license agreements are enforceable. Speaking through Judge Easterbrook, the Seventh Circuit examined applicable copyright law, Article 2 of the U.C.C., and the commercial reality of the mass-market software transactions. The court discussed the commercial justifications for the contract and rejected arguments that the contract was procedurally unconscionable.

ProCD continues to be a controversial decision.[108] There have been subsequent decisions which do not follow this decision although most courts ruling on the enforceability of mass-market licenses post *ProCD* have held them enforceable providing they follow procedural requirements.[109] In general, for a mass-market software license to be enforceable, the software consumer must be given three things: proper notice of the license before purchase, adequate time to review and decide whether to assent to the license's terms, and the opportunity to return the software for a full refund if the license is unacceptable.[110] Whether or not a Canadian court would adopt a similar test remains to be seen. If a court analogized an open source license to that of a clickwrap license, the enforceability of the license would be unclear. Because a minority of U.S. courts still hold shrinkwrap licenses unenforceable while Canada has yet to rule on the issue at all, the enforceability of shrinkwrap software licenses remains uncertain.

Although similar to shrinkwrap and clickwrap agreements, open source licensing differs from these licenses in that they contain many novel substantive provisions. For example, open source licenses place requirements and restrictions on the licensees who wish to modify earlier versions of the software. The GPL license states, "by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it".[111] Such a provision may even extend beyond clickwrap and shrinkwrap provisions.[112] The enforceability, therefore, of open source licensing will depend on whether it complies with general contracting principles and on the interplay between applicable statutes governing software licenses.

The conclusion will argue that the *Copyright Act* should be amended to alleviate the legal uncertainty surrounding the validity of open source licensing.

Trademark

The open source movement makes sophisticated use of trademark law. One approach for the community would have been to register a trademark for the software products and distribute it under that name. Such an approach would have, however, raised trademark law issues with the open source model of distribution. Open source software can be freely adapted and distributed further by people other than the original producers. Under the law, a trademark must identify the source of goods.[113] If adapted and redistributed software bore the original mark, it would be a misleading use unless everyone producing open source banded together as a single producer. Additionally, the trademark holder must police the use of the mark. Just as a trademark holder cannot make a blanket assignment, they also cannot allow others to use the mark without verifying that their goods or services conform to their standards. Thus, typ-

ical use of trademark in the open source situation would have led to trademark problems down the road. The open source movement, however, incorporated a more refined use of trademark, the certification mark. Unlike most trademarks, the mark is not used by the owner, but rather by others to indicate that it meets standards set by the mark owner. These certified marks are used to signify that a manufacturer or producer has complied with the relevant standards. The Open Source Initiative decided to register a mark that it would permit others to use if their software complied with the Open Source Definition.

The initial mark chosen, however, violated trademark law. The first mark chosen was "Open Source" using a recently coined term that succinctly described the movement to make source code freely available.[114] The apt nature of the mark made it questionable as a matter of trademark law. The open source developers sought a name that was as descriptive as possible, therefore, they opted to register the mark, "Open source".[115] It soon became clear the United States Trademark Office ("USTO") would likely reject the marks on the grounds of descriptiveness because under trademark law a trademark cannot be registered if it is simply descriptive.[116] This left open several possibilities for the Open Source community. They could have registered, "Open source," then challenged the USTO's decision to invalidate the trademark because it was descriptive. They could also have registered the mark on the supplemental register, then have applied for a principal registration after it had sufficient publicity to acquire a secondary meaning. The open source initiative, however, decided simply to change trademarks by registering the certification mark, "OSI Certified".[117]

OSI Certification allows the open source community to control and maintain the integrity of its development model, acting as a safeguard against the manipulation and unwanted alterations of the system. Certification can be seen as an additional control mechanism to licensing which allows the open source community to privately "regulate" the dissemination of open source software.

Patents

The attributes of open source licensing rest primarily upon the license protections provided by copyright and trademark law. Most open source licenses do not specifically address the issue of patents, but open source developers are affected by patents as possible inventors or infringers. An open source software program must be sufficiently novel and inventive as to be patentable in the United States. This situation in the United States must be juxtaposed with the Canadian situation. While computer software is patentable in Canada, patenting software has not been prolific unlike the situation in the United States. Open source software development in Canada, therefore, is not currently threatened by the proliferation of Canadian software patents. That being said, open source software is primarily developed through communication that occurs over the Internet. Contributors to projects may come from all over the world. The likelihood of an open source project being limited to an all Canadian collaboration is slight. Moreover, the impetus of many open source initiatives either stem from or involve contributors from the United States. U.S. patent law, therefore, remains very relevant to open source development.

Authors of open source software may wish to patent their works in order to make money by licensing the patent or they may wish to patent the process to make sure no one else did thereby keeping the process free for public use. The patent gives the holder the exclusive right to use his or her process, or to make or sell a machine containing the invention.[118] By distributing the software under an open source license, the author would authorize others to use the program free of his or her patent claim.

At the other end is the risk of infringing someone else's patent. Even if an open source author devises his or her program independently, its use could infringe a patent he or she did not know to exist. The problem arises when someone copies or uses the open source software thereby becoming liable for the use of a patented software without having paid the appropriate fee to the patent-holder.[119] The user could then turn around and sue the open source author for infringement of the warranty of good title, especially if the user had paid for the copy. The blanket disclaimer warranty, however, similar to that of the GPL, may protect the author, as could a number of other arguments depending on the circumstances. Such mitigating circumstances would be whether the author charged for the copy, whether he or she was a software merchant, or whether the program was consumer software.[120]

Some individuals think that software patents may pose the greatest threat to open software.[121] After considerable reduction in the legal obstacles to patenting software, many thousands of software patents have been issued in recent years in the United States.[122] Open software developers might write code that allegedly infringes such patents. Another commentator has described software patents as "a minefield for open source developers".[123]

Open source software defendants will have one particular disadvantage as compared to other software developers who might be potential patent infringers; this risk arises from the very nature of open source software.[124] Suppose someone holds a patent on a process used in software — a process for sorting data, or for producing a particular format of output. If a proprietary program used the patented program, the patent holder might not be able to discover the use. The process might be used in the program, but not in a way that was

evident to a user of the program. One could tell that the program was, at some point, sorting data, but would have to go to considerable trouble to figure out how the program was sorting it. Indeed, that would be impossible if one did not have access to a copy of the program. It would be much easier to monitor open source programs for infringement of the patent due to the very model on which open source is premised. One would be entitled to obtain both a copy of the program and a copy of the source code making open source peculiarly susceptible to patent monitoring.

Another area in which open source developers could be at a disadvantage is in cross-licensing. Because so many software patents have been issued in recent years, and perhaps because the validity and enforceability of many of the patents is rather unclear, patent licensing is quite different in the software area than in other high-tech areas such as biotech.[125] In particular, royalty-free cross-licenses are quite common in the computer industry.[126] The parties to such licenses agree, in effect, not to attempt to enforce their patents against each other. Such non-aggression pacts protect only the parties to the license. To the extent that open source developers do not seek software patents, it may leave them unprotected, having nothing to offer in return.[127]

Open source developers may, however, have the advantage of being able to redirect the course of software patent litigation.[128] A significant issue in software patent law is the problem of prior art, that the invention is both novel and non-obvious — that it would not be obvious to a skilled member in the field.[129] Locating prior art in the computer software field is difficult. The patentability of software is a somewhat recent development. Therefore, the stock of software patents to provide a source of prior art is limited.[130] Additionally, computer programming does not have systematically archived knowledge.[131] One commentator has determined that 80 per cent of issued software patents make no effective citation of prior art, despite the great amount of published work in computing.[132] A defendant, therefore, in a patent infringement action in the United States may have an extremely difficult time proving that a technique was already in the prior art. Open source developers, however, present a formidable resource for locating prior art due to the very nature of the software, that the source code is open and, therefore, more readably accessible to demonstrate prior art. As in the case of copyright, open source developers are in the situation to posit themselves as a counter-balancing force against the over-patenting of computer software programs.

Benefits of Open Source Development

Competition Against the Market Dominance of Traditional Players

For the computerized world to continue turning, computers must be compatible. This quest for compatibility led to industry standards in architecture on the hardware side and increased market domination on the software side. For example, Mircrosoft's MS-DOS (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 many compatibility concerns, but the domination ultimately triggered an antitrust action against Microsoft.[133] Open source avoids antitrust or competition issues because no single entity owns the code. More importantly, open source software, especially the Linux system, offers competition to those dominant players in the computer software industry.

There has been speculation that Linux, an open source initiative, may be in a position in the near future to "de-throne" Microsoft in the operating system market, or some substantial portion thereof.[134] The open source community, especially Linux, is garnering greater support for its products. Linux represents a legitimate alternative to Microsoft. Microsoft's recent antitrust problems have resulted in resentment for Microsoft's anti-competitive conduct. Although not a necessary trait of the open source community, there seems to be an impression that anti-Microsoft sentiment plays a role in the interest and participation of some in the open source movement.[135]

Microsoft executive, Jim Allchin, believes that open source software, particularly the Linux operating system, "stifles and … threatens innovation".[136] What open source projects really affect is market domination. A more accurate description would read, "open source stifles and threatens profits". The open source model, as many programmers in the open source community have said, will never replace proprietary models of software development altogether. These same software programmers, such as Eric Raymond and Linus Torvald, readily acknowledge that each software project should be tailored to the system it is best suited to; sometimes this will require using an "open" system and at other times, a "closed" system.[137] Open source will, however, offer a viable and competitive alternative to the software and operating systems currently offered by the dominant

industry player(s). As a competitive market is generally understood in both the Canadian and American economy to be beneficial to society, and is a fundamental construct to any market economy, the competition that open source programming brings to the arena should be welcomed at least, and readily encouraged at best.

Benefits of a Superior Model: The Economics of Efficiency

Economically, open source has been seen as 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.[138] Open source exhibits valid, economical, and marketable alternatives to proprietary software development and distribution.

Open source programmers are proud to extol what they believe to be a superior method of software development. It has been argued, for example, that open source projects can produce better quality technology than traditional corporate research and development.[139] The open source model operates on the premise that, by having the opportunity to build on each other's ideas, rather than duplicate one another's efforts in a "closed" system, software developers are able to produce more efficient, and technologically superior products. This efficiency and superiority stems from a few basic principles as expressed by Raymond: many heads are better than one, and people are most motivated when they are personally interested in the work.[140] An open source project employs the Internet as its means of making the source code available attracting a potentially unlimited amount of co-developers. These co-developers are able to look at the code, improve it, make suggestions, locate bugs, debug, and so forth much more rapidly than in a "closed" system. An open source project can be seen as a community of problem solvers, co-developers, and ultimately, consumers of the product.

The great difference in economic analysis of proprietary software development and open source development is well illustrated by the topic of network economics and industry standards. For example, Mark Lemley and David McGowan discuss why for-profit software companies might not have the incentive to develop a potentially huge-selling product such as the Windows operating system, namely, because such products would be protected by copyright law.[141] Copyright law only protects the expressive aspect of works, not their functional aspects.[142] So another software company could copy the unprotected functional aspects of Windows and sell the functionally equivalent program to a big market without violating the copyright. As Lemley and McGowan explain, several reasons stand in contradiction to such a strategy.[143] Although copyright does not protect functional aspects of the program, there remains legal uncertainty about which aspects are functional.[144] Also, other intellectual property, such as patents or trade secrets, might protect some aspects of the program. More important than the legal uncertainties, perhaps, the market risks would be great deterrents to a commercial competitor. Reverse engineering[145] the program is a time-consuming and uncertain enterprise, and Microsoft periodically upgrades the program, which means that a commercial competitor might have difficulty in selling an up-to-date product.[146] Consumers might also be wary of whether the program was truly a functional substitute. Finally, Microsoft presumably has the ability to lower the price of its program to compete with any new entrant, so the potential payoff is greatly reduced. All in all, it makes little sense for a commercial competitor to make the huge investments in development and marketing that would be required to compete when other avenues of investment are likely to be more fruitful.

The economic incentives for open source developers are quite different. Certainly some open source developers simply wish to sell software, and would thus be subject to the same disincentives. As we have already seen, for many open source developers the incentives are quite different: enjoyment of programming itself, the desire to show off technical feats to others in the field, the wish to sell software-related services, an idealistic urge to further computer science, and even the desire to tweak the proprietary software companies. Nor would they be scared off by the fact that an existing software product seller could respond to a new rival by lowering prices because the open source developers are giving their product away. Therefore, open source developers might be willing to take on a task, such as building a Windows emulator, where a profit-seeking enterprise would not. Indeed, just as economic theory might predict, such an enterprise exists: the WINE project[147] is an open source project building a Windows emulator to run with the Linux operating system (a piece of software that copies the functionality of Windows running on a Linux system).

The premise that open source is not commercially viable is false. Open source operates on unique and novel business models. The head of the Open Source Initiative, Eric Raymond, offers the following means of recuperating investment: (1) market positioner/loss leader;[148] (2) widget frosting;[149] (3) give-away recipe/open restaurant;[150]; (4) accessorizing;[151] (5) free the future, sell the present;[152] (6) free the software/sell the brand;[153] and (7) free the software/sell the content.[154] In summary, the open source development model is beneficial in many ways: it is seen by many as a superior development model; it fosters innovative projects that would be too risky for proprietary software; and it provides new business models for the rapidly changing software industry.

Balancing Factor Against Over-Expansion of Industry and Government Control: Protecting Fundamental Societal Values

In addition to fostering the creation of standards that increase societal wealth, open source can be seen as a critical balancing factor. Recent articles have posited that open source is a threat to copyright, that it undermines innovation, and is becoming a threat to viable copyright industries. In reality, open source software acts as a counter-force to an increasingly unbalanced system. As already demonstrated, open source fosters competition in the software industry while at the same time, it offers an alternative development model which many see as superior to a "closed" model thereby encouraging and promoting innovation at an increased rate. But more importantly, the open source community is representative of a democratic force acting against the encroachment of market domination in not only copyright, but in the control and architecture of the Internet. The societal values at stake are imperative.

Arguments about open source must also consider the values at stake and the issues raised aside from economic wealth and efficiency. Social and cultural values may be impinged by the ownership of source code. The control and development of source code, in turn, substantially impacts on many values that society holds as fundamental to a free and enlightened democracy.

Open source is about challenging boundaries, re-examining notions of how property is held and how wealth is distributed in our society — economic, social and cultural wealth. There is currently a change of the power paradigm in North America over who will control the Internet and, perhaps to a greater extent, the content which is conveyed over the Internet.

The greatest threat is not the government. So far the government has regulated very little of what happens or what is conveyed on the Internet. Increasingly, it is the market itself that is indirectly being handed this power, the power to control the architecture of the Internet.

Open source acts as a check to the potential abuse and dominance by market players as they are given more and more tools with which to control the Internet. The situation is, for the moment, somewhat different in the United States than it is in Canada. Firstly, the owners of the underlying architecture of the Internet, that is the wires and cables of connectivity, do not currently possess the same controlling powers in Canada as their American counterparts.[155] In the United States, AT & T has been given the tools and leeway not only to develop the underlying architecture of the Internet, but more importantly, the power to control and dictate the content of what they will allow or disallow to be disseminated.[156] Thus, we are seeing a steady increase in the power and control of both content and software over the Internet given to the "big players".

Open source operates under a transparent model. As Lessig states:

> A transparent modularity permits code to be modified; it permits one part to be substituted for another. The code then is open; the code is modular; chunks could be removed and substituted for something else; many forks, or way that the code could develop, are possible. No one dictates which way the code will develop … Instead, the evolution of the market does that. The evolution of thousands of people trying their hand at improving a code, and thousands of people choosing which improvement makes sense. The consequences of the open-evolution design … is that no one can control how the system will evolve. No single individual gets to set the path the system will follow. It might evolve to follow a path, but it will evolve by the collective choice of many.[157]

The fact that you or I are free to take a source code that is open, modify it and improve it without the permission of Microsoft or the government is not important at a practical level for the average citizen, but it is representative of a kind of formal equality between "the little guy" and "the corporate giant". Moreover, it is the recognition of an imbalance in the system from a bottom-up approach. An imbalanced approach may lead to the privileging of private values at the expense of displacing public values.[158] Open source acts as a force against the potential for imbalance.

Before we can examine any imbalance and arrive at a plausible conclusion, we must ask a fundamental question — why does software matter? We are witnessing the exponential growth of a new form of social structure known as the information society. The quintessential element of discourse, of language, and of speech in this information society is software. Software is now a key part of our social structure — it is in our cars, supermarkets, televisions, and computers. As one academic describes software, "We sense it everywhere: it is a ubiquitous, undulating, architectural, air-like, water-like commodity that infiltrates our daily lives".[159] Even more interesting is that software, through its various forms of coded structure, can act to construct meaning and identity in much the same way as speech. Some would even go for as to say that software is discourse.[160] Software is not simply a literary text subject to copyright law protection; it is a form of discourse and a fundamental tool in democracy. Thus, the debate over open code versus proprietary code software is intimately linked to the notion of the construction and moulding of society.

The legal and customary regimes that control software dictate which values are promoted in society. This may, in turn, lead to an imbalance in power. To illustrate the dangers inherent in an imbalanced regime, we will look to the over-breadth of copyright law by examining anti-circumvention measures.

The two traditional competing approaches to copyright law are the neoclassicist approach and the democratic paradigm approach.[161] The former emphasizes that

copyright has as its primary goal, the promotion of allocative efficiency. The latter contends the basis for copyright doctrine is the support of a democratic culture.[162] Copyright doctrine, in Canada, looks to both of these notions. For the purpose of this example, however, we will only be concerned with the democratic paradigm. For the purpose of the anti-circumvention example, we will only address the democratic aspects of copyright.

In the digital age, works of copyright are often protected through the use of encryption. Encryption technologies scramble code acting as protective system to avoid unauthorized access. Circumvention means to avoid the effect of a technological measure designed to prevent unauthorized access to a system or mechanism such as a database, satellite system or a DVD movie. In 1998, the United States Congress enacted an anti-circumvention provision in the *Digital Millennium Copyright Act*.[163] The Act imposes civil and possible criminal liability for the circumvention of access control measures and for the distribution of technology to circumvent access controls. Although Canada has not yet enacted similar circumvention provisions, they are expected to do so in the near future.[164] Anti-circumvention provisions make it illegal to crack a protection regime, even if the use of the underlying material is not itself a copyright violation.

The U.S. legislative provisions on digital technology have been in place for nearly five years. During this period, the DMCA has been heavily criticized for its over-breadth, for its encroachment on free speech, and for its impact on fair use. Critics, such as Benkler, argue that the DMCA and other laws that purport to protect information as a commodity, are over broad, the end effect of which is to remove uses of information from the public domain and place them in an enclosed domain where they are subject to an owner's exclusive control.[165] Exclusive control is accomplished through the utilization of technological protection measures such as encryption. Under the DMCA, circumvention of a technological protection measure such as an encrypted code or a digital rights management system, is a separate legal wrong from copyright infringement. To circumvent a technological protection measure, regardless of whether copyright infringement has occurred, is to break the law under the DMCA. For example, private parties can use a rights management system to determine the rules that will be embedded into technological controls. Rights management systems allow copyright holders to restrict uses of their works. Perhaps a song will only be allowed to be listened to twice. Or the private copying of a work will not be allowed. Even still, perhaps a CD will only play on certain authorized machines. It is easy to see how such measures may have crippling effects on users impairing fair uses to copyrighted works.

The impact of a copyright holder's exclusive control through technological protection measures is more devastating when coupled with anti-device measures. The prohibition on circumvention allows an organization or company to control the legitimate use of a work by erecting an encryption code. It has been further demonstrated how this type of prohibition could have the stifling effect of denying access to works that are in the public domain or works, which would normally fall under an exemption or, as fair use under the current copyright regime. By prohibiting users from possessing the decryption tools necessary to break such technological protections, the DMCA effectively promotes "digital lock-up". In other words, you may have a legitimate reason or legal right to use a copyrighted work that is protected by a technological protection measure. This right is, however, rendered utterly ineffective if a user does not have the right to possess a device capable of circumventing the technological protections.

Others, such as David Nimmer, describe the DMCA as a philosophical tug-of-war with industry players wishing to protect copyrighted works at one end with the public on the other end. In his analysis, Nimmer entertains the notion of a pay-per-use world emphasizing the need to ensure that there is a balance between property rights and access to works founded on public policy sufficient to, "rise to [a] constitution level".[166] As it currently stands, the circumvention provisions provide a new access right, a right unknown to copyright prior to the DMCA, skewing the balance in favour of copyright holders.

Jane Ginsburg, on the other hand, articulates that the real problem does not arise from the fact that an infringer must pay for initial access, but primarily when they cannot obtain continued access on reasonable terms.[167] She highlights that access control measures may not be intended to prohibit scholarly or critical examination of the works themselves but this may be the result if the user cannot consult or acquire a fair-usable copy at a reasonable price or from a public source.

Professors Benkler and Lessig further note, in an *Amici Curiae* brief to the New York Court of Appeal in the *Reimerdes* case,[168] that the anti-device provisions of the DMCA place a burden on technologically unsophisticated users, and seriously undermine copyright exemptions such as fair use and reverse engineering. By way of background to this case, a Norwegian teenager, Jon Johansssen, developed the software DeCSS that enables users to break the CSS copy protection system allowing free distribution and viewing of DVD movies over the Internet. He did so by reverse engineering the CSS encryption algorithm, then writing a program which decrypts CSS. He then posted this program, DeCSS, on the Internet to notify other programmers that DeCSS ran on Linux platform (CSS does not run on the open source platform, Linux). Universal Studios initiated several lawsuits; one of which involves three individuals, Shawn Reimerdes, Roman Kazan and Eric Corely who are associated with the operations of *2600*, a magazine for hackers. DeCSS was posted in this magazine. In spite

of the fact that DeCSS was developed to run on the Linux platform and was done through the lawful practice of reverse engineering, the U.S. District Court of Southern New York granted an injunction prohibiting both the posting and linking to Web sites containing DeCSS. The Court further held that DeCSS was a means of circumventing a technological access control measure under the DMCA, and was therefore a prohibited act. The Court focused its decision on one particular function of DeCSS — its ability to enable distribution of DVDs over the Internet or which they labelled, piracy. The Court was quick to dismiss the arguments that DeCSS was developed according to copyright exemptions, that of reverse engineering a program to make it operable on another operating platform (Linux in this case). The mere posting of DeCSS on a Web site was to contravene the DMCA. The decision was appealed and subsequently dismissed on appeal. Commenting on the case, the authors of the *Amici Curiae* note that the problem is not deterrence but incapacitation, and that the DMCA unduly restricts freedom of speech. They emphasize that the millions of regular users who are not computer professionals, and whose ability to express themselves in creative and meaningful ways is severely and unnecessarily undermined by the provisions of the DMCA. Furthermore, by limiting the scope of reverse engineering and other fair uses, the vitality of open source development projects may be hampered. It is, however, too early to fully ascertain the effects of this decision on the open source community.

On the other hand, open source software, albeit in a limited capacity, also stands in opposition to the overbreadth of copyright such as anti-circumvention measures. Open source represents a model that encourages users not only to freely access and copy the program but also to modify it. Where the DMCA and similar anti-circumvention measures act to displace values in code, open source responds by reclaiming the values displaced. Thus, in essence, open source code programmers are neutralizing the potentially harsh effects of the anti-circumvention law. Open source neutralizes some of the effects of anti-circumvention measures through its promotion of free speech, sharing of ideas, the building of new ideas upon old formations, and by letting the users of the code determine and control the manner in which the programs will develop. This is not to say that open source counterbalances the negative effects of anti-circumvention provisions, but rather, it is representative of a small counteractive force.

Now consider the ability of open source to counter state control and regulation of source code. Open source checks governmental power by limiting the extent that the Internet may be regulated and by acting as a countering force where the government has over-regulated.[169] This is because open source code is more difficult to regulate than its closed source counterpart.[170] The very nature of open code is that it can be manipulated and altered as the user wishes. As Lessig states, "Open code means open control — there is control, but the user is aware of it".[171] As the user is aware of what the actual code entails, he or she can ultimately choose to utilize the software or to utilize part of the software. The state may impose regulations on the source code, but the end user is able to decipher this control, and thus, is in a better position to filter and tailor the code. To cast the regulative nature of open source in a different light, it is difficult to regulate and control anarchy; the anarchy is a means and an end to itself.[172]

Closed code, on the other hand, functions differently. With closed code, users cannot easily modify the control that is inherent to the code. Hackers and sophisticated programmers may be able to do so, but most users would not be able to see which parts were required and which parts were not.[173] Thus with closed code, required elements and features may be hidden from its users.

The government's power to regulate code depends on the character of the code. Open code is less regulative than closed code; to the extent that code becomes open, government's power is reduced. This is not to say, however, that reducing governmental power to regulate is necessarily a good thing. It may be desirable for the government to regulate source code in certain circumstances. Where and when there is a general consensus that it is desirable for the source code to be regulated, it is likely that users will choose not to modify those aspects of an open source software — for it is the user that has the option of selecting in an open system. Open source will, however, be a beneficial counterforce where over-regulation or unwanted regulation has occurred.

Fostering Open Source

Should intellectual property laws be amended or interpreted in order to foster open source? Given the important and invaluable economic and social role open source plays in the computer software industry and the impact that computer software has on shaping and validating values in a democratic society, legislation and regulatory measures should be developed to promote and encourage open source.

Open source software offers many puzzles and lessons for intellectual property theory and doctrine. Considerable legal scholarship in recent years has lamented the increases in intellectual property protection that have steadily diminished the public domain,[174] but the open source movement has countered this trend. The open source movement has used strong protection of intellectual property to quite different ends. In particular, various open source licenses rest on strong copyright protection and restrictive licensing provisions. However, open source licenses use such restrictive law to keep open source code free. Because intellectual property laws place so much control in the hands of copyright owners, various flavours of open source licenses are able to finely

tune the way in which code is kept available for others to study, modify and redistribute. That does not mean that open source should provide a justification for the overbreadth of intellectual property protection. The boom in software patents, as we have seen, is a considerable threat to open source.

The reasons for encouraging open source software parallel arguments made to amend the *Copyright Act* to include a reverse engineering provision. Proponents of the inclusion of a reverse engineering provision in the early 1990s articulated that reverse engineering would be consistent with the underlying policy goals of copyright law within an economic/utilitarian framework and the promulgation of such a provision would foster the creation of standards that increase societal wealth.[175] Similarly, open source software is consistent with the goals of copyright both from an economic/utilitarian approach and from a moral rights approach.

An economic/utilitarian approach to intellectual property is rooted in the belief that financial incentive is necessary to spur the creation of works. Intellectual property laws aid creators to be compensated for their works. The compensation for work is said to encourage and foster more creative works. Open source, at first glance, does not provide for an economic benefit to its creators but this does not mean that there is a lack of incentive to create. The incentives are simply different than financial reward. As we have seen, reputation, the act of creating itself, along with a sense of creating a superior product provides ample incentive to open source programmers. In this respect, open source is consistent with the economic/utilitarian model.

In many ways, open source embodies the underpinnings of the moral rights approach. Moral rights in copyright works stem from the belief that creations emanate from an individual's personhood. The creation of a work is intricately linked to the human body and soul, and has been associated with a human right. Moral rights in works exist to ensure that the artistic integrity of a work is not compromised and that the reputation of the artist is not degraded. The terms of open source licenses ensure both of these qualifications. The only substantial difference is that the artist or author of an open source project is not one singular person, but the entire team working on the project. While there is often a team leader, credit for the work belongs to the entire team. It would be difficult, if not impossible, to denigrate the integrity of the author through modification of the source code when the very purpose of open source is to do just that, to modify the source code in order to improve upon the project. As we have already seen, the chief incentive of open source projects is the enhancement of reputation. Open source licenses ensure that credit is given to those who work on projects, and this credit helps to ensure that the reputation of the team members is not tarnished.

Some argue that trying to interpret or enact intellectual property laws to foster open source may well be beyond the planning of government.[176] I argue that it is not.[177] The government cannot ensure that open source development remains active but they can enact legislation which would, at a minimum, remove the ambiguities surrounding its legal validity.

Such legislation might entail pronouncing the legal validity of mass-market licenses such as those incorporated by the open source movement. Legislation which clearly pronounced the legal validity of mass market licenses, with particular reference to warranties, would alleviate the uncertainty and reservations that many businesses currently have of using open source software. Canadian codes for e-commerce legislation could likewise provide for a successful list of criteria in the validation of shrinkwrap licenses, and could take the additional step of outlining criteria for the validation of open source licenses. This would provide guidance for a court should the validity of an open source license find its way to litigation.

The most important change, however, will come in the choice that the Canadian government makes with respect to copyright reform. Should Canada adopt a framework similar to the DMCA, the vitality of the open source community may be at risk. More specifically, the adoption of an anti-circumvention measure, without a strong articulation and support of robust and flexible exceptions, could impact on future open source development projects. Canadian copyright law currently contains a fair dealing exception to copyright, when a work is used for the purpose of private study, research, reverse-review, criticism, or news reporting and the manner of the use is fair.[178] A fairly long list of other specified exceptions to copyright exist in Canada in order to protect educational institutions,[179] libraries, archives and museums,[180] computer programs (reverse engineering),[181] incidental inclusions,[182] ephemeral recordings,[183] and sound recordings.[184] It is imperative that any reform made to the *Copyright Act* continue to support the current copyright exemptions. For the open source community, the key exemption is the ability to reverse engineer computer software programs. Without such a defense, it may curtail programmers from producing compatible software to run in the Linux platform without infringing copyright.[185]

Additionally, some government agencies could use open source software to help reduce concerns about the products. These government agencies could serve as a test site to show large corporate users that open source software is viable. The result of large-scale government involvement in open source could calm many of the marketplace concerns regarding open source products. Likewise, Industry Canada could commission a study on the reliability and security of open source applications,

and the relevant costs associated with switching platforms.

Governmental groups in Canada are slowly beginning to turn their attention to open source software. The Quebec provincial government has produced a formal study of open source software.[186] Several open source symposiums have been recently held including: Ottawa Linux Symposium,[187] Bioinformatics Open Source Conference in Edmonton,[188] while National Defense, Industry Canada and Heritage Canada are beginning to look at Linux.[189] These efforts, however, are very much at the preliminary stage. Pro-active measures to ensure the viability of open source in Canada remain to be taken. This author strongly recommends that Canada look to the pro-active measures that other nations have taken recently to foster open source software.

The German government has taken positive steps to foster open software development.[190] The Federal Ministry of Economic Affairs intends to publish a guide this fall that will be targeted at SMEs and the civil services. Its aim is to clarify the advantages and disadvantages of open source software and should serve such institutions as an introduction of the subject. A national open source competency center will also be set up to serve as a nodal point for the open source community and software users in Germany, by providing the necessary technical infrastructure, a discussion forum and marketplace. Additionally, the German government is switching its operating platform from Microsoft to Linux.

The South African government is considering taking similar measures to Germany to help foster open source development. The National Advisory Council on Innovation released a working draft paper entitled, "Open Software and Open Standards in South Africa."[191] The report recommends the broad-scale promotion of open standards and open software in the public sector and in business. The document highlights that the long-term benefits of opting for the open source route will be significant. The development of open source skills is seen as an independence mechanism whereby South Africa will not have to rely on foreign skills for software development.[192] According to the discussion paper, open source software, "has the potential to empower people in ways that proprietary software simply does not allow".[193] While the South African government has yet to adopt the recommendations put forth in the policy paper, they are expected to endorse many of the recommendations from the policy study. Furthermore, the release of this discussion paper represents a growing trend towards open source development.

A new threat to the functioning of open source is the proposed *Security Systems Standards and Certification* (SSSC) Bill in the United States.[194] This Bill will make it unlawful to manufacture, import, or traffic in any interactive digital device that does not include and utilize certified security technologies. Critics argue that the Bill, as currently written, would prohibit the Linux kernel and many software programs developed to run on Linux.[195] Open source poses very few restrictions on its developers in order to remain innovative and "free" in nature; the use of standardized technologies would go against the very essence of the movement. Such programming restrictions have been seen by the open source community as having the effect of stifling the progress of software development. Hopefully, modifications to the Bill will allow the continued growth of open software. It is further hoped that Canada will not elect to enact legislation similar to the SSSC.

Conclusion

What does open source teach us? That the free movement of ideas is a fundamental tenant to the preservation of open society. That balance is everything. That innovation is not always motivated by profit.

Open source is a movement that asks us to re-examine the validity of the economic and philosophical underpinnings of intellectual property in the digital era. It is a movement that asks us to consider alternative business models and approaches to software development. And it is a movement that acts as a counterforce to the encroachment of industry and government control and power over code.

The legal validity of open source is an open-ended question in Canada. There has been no articulation of required conditions for an open source licensing agreement to be valid. While open source licenses parallel clickwrap and shrinkwrap licenses, a Canadian court has not had the opportunity to pronounce on whether this will indeed be the adopted analogy. Clickwrap licenses are valid in Canada but the validity of shrinkwrap licenses remains unclear. Where the validity of shrinkwrap licenses have been litigated in the United States, the courts have rendered divergent decisions.

The vitality of open source is also threatened by copyright and patent law in the United States. Although Canada has not embraced software patenting in the prolific manner of the United States, patenting still threatens the movement to some degree. The extent that copyright law undermines the vitality of open source will largely depend on whether the Government adopts DMCA-like anti-circumvention legislation. Anything diminishing the scope of reverse engineering of a software program will impact on the open source community.

Open source is indeed an open-ended question. I propose that society embrace this movement with open arms.

Notes:

[1] The term "copyleft" and "freeware" connotes a more revolutionary stance of software developers. Such developers feel that protection of software through current intellectual property regimes is ethically wrong and an impediment to innovation. This movement is led by Richard Stallman, founder of the Free Software Foundation, *infra* notes 3 and 4. Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. The "free" connotes freedom and not price. The term "open source", on the other hand, represents two separate notions. Firstly, the term "open source" has been associated with those developers with a less radical vision; they do not share the same ethical dilemmas of proprietary software and are often attracted to open-source development as a superior method. "Open source" is also considered the umbrella term for non-proprietary software development which incorporates the varying philosophies of the movement, *infra* note 15.

[2] Source code is the text of the program written in a programming language which is easily discernable to computer programmers. Object code, on the other hand, is instructional data run directly on a computer's processor which is written in a string of 1's and 0's.

[3] See Richard Stallman, "A Serious Bio" found at <http://www.stallman.org/#serious> (date accessed: September 9, 2002).

[4] See, "What is Free Software" found on <http://www.gnu.org/philosophy/free-sw.html> (date accessed: September 9, 2002).

[5] This famous quote from Richard Stallman has become somewhat of a symbol for Free Software Foundation. See Stallman, "Think Free Speech, Not Free Beer" at <http://www.gnu.org/philosophy/free-sw.html> (date accessed: September 9, 2002).

[6] Richard Stallman, "The GNU Project" at <http://www.gnu.org> (date accessed: September 9, 2002). GNU is not Unix; it is a Unix-like operating system. "GNU" is a recursive acronym for "GNU's NOT Unix".

[7] Stallman, "Initial Announcement" at <http://www.gnu.org> (date accessed: September 9, 2002).

[8] *Ibid.*

[9] Free Software Foundation, "The GNU Manifesto" at <http://www.gnu.org/gnu/manifesto> (date accessed: September 9, 2002).

[10] See Technology Encyclopedia at <http://www.techweb.com/encyclopedia/> (date accessed: September 9, 2002).

[11] S. Potter, "Opening up to Open Source" (2000) 6 Rich. J.L. & Tech. 24.

[12] Stallman, "GNU Project", *supra* note 6.

[13] *Ibid.*

[14] R. Gomulkiewicz, "How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B" (1999) 36 Hous. L. Rev. 179 at 184.

[15] E. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (Sebastapol: O'Reilly & Associates, R'vd Ed, 2001) at 66.

[16] See G. van Rossum, "Open Source Summit Trip Report", online at <http://www.linuxgazette.com/issue28/rossum.html> (date accessed: September 9, 2002); and K. Porterfield, "Information Wants to be Valuable: A Report from the First O'Reilly Perl Conference (1999)" on <http://www.netaction.org> (date accessed: September 9, 2002).

[17] T. O'Reilly, "The Open-Source Revolution, Release 1.0", at <http://www.opensource.org> (date accessed: September 9, 2002). The open source Web site provides a plethora of software programs designed for the Internet.

[18] *Ibid.*

[19] *Ibid.*

[20] *Ibid.*

[21] Raymond, *supra* note 15.

[22] *Ibid.* The superiority of the open source model is discussed at length in the section, "Benefits of a 'Superior Model': The Economics of Efficiency".

[23] *Ibid.* See also I. Tuomi, "Internet, Innovation, and Open source: Actors in the Network", (2001) available on <http://www.firstmonday.dk/issues/issue6_1/tuomi/> (date accessed September 9, 2002).

[24] M. Maher, "Open Source Software: The Success of an Alternative Intellectual Property Incentive Paradigm" (2000) 10 Fordham Intell. Prop. Media & Ent. L.J. 619.

[25] Gomulkiewicz, *supra* note 14 at 185.

[26] T. Hill, "Fragmenting the Copyleft Movement: The Public Will not Prevail" (1999) Utah L. Rev. 797 at 799.

[27] *Ibid.*

[28] Stallman, *supra* note 4. A complete philosophy of free software is outlined at <http://www.gnu.org/philosophy>.

[29] *Ibid.*

[30] *Ibid.*

[31] Hill, *supra* note 26 at 801.

[32] Raymond, *supra* note 15 at 68.

[33] Stallman, *supra* note 3.

[34] *Ibid.*

[35] *Ibid.*

[36] "Free" refers to the freedoms associated with non-proprietary software development. Thus free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software.

[37] Raymond, *supra* note 15 at 69.

[38] *Ibid.*

[39] *Ibid.* at 70.

[40] Tuomi, *supra* note 23. The idea of communities of practice has also received attention in the context of innovation theory. See also, J. Brown and P. Dugoid, *The Social Life of Information* (Boston: Harvard Business School Press, 2000).

[41] Raymond, *supra* note 15. Raymond compares the incentives and customs of the open source community to other "gift cultures" such as the Kwatkiutl and Trobriand Islanders.

[42] The linking of the open source movement with gift cultures has been criticized as an inadequate exploration of the driving forces behind the movement. Open source programming, according to some, is additionally motivated by economic incentives; economic in that early funding for this software was a publicly-subsidized market motivated by the immature state of the software market. See D. Lancashire, "Code, Culture & Clash: The Fading Altruism of Open Source Development" available at <http://www.firstmonday.dk/issues/issue6_1/>. Others see reputation as a form of currency where, "currencies are equally capable of expressing value, and could be used to reckon a network of obligations, perhaps even converted into each other". See C. Kelty, "Free Software/Free Science" available at <http://www.firstmonday.org/issues6_12/>.

[43] Maher, *supra* note 24.

[44] *Ibid.*

[45] It is important to note that the gift-culture idea *may* be counter-intuitive to businesses. This, however, would not address the prevalence of corporate philanthropy. Corporate philanthropy may be contrasted with the open source movement. A company may donate funds as a marketing plan done to enhance the reputation and image of the company. Such philanthropy is still premised on profits — an enhanced reputation will yield greater sales. Additionally, corporate philanthropy is often motivated through taxation deductions. Reputation within the open source movement does not resemble corporate philanthropy in either of these respects. For further discussion of philanthropy see, S. Rose-Ackerman, "Altruism, Nonprofits, and Economic Theory" (1996) 34 J. Econ. Lit. 701.

[46] R. Young, "Giving Away" in C. DiBona, S. Ockman, and M. Stone, *Open Sources: Voices from the Open Source Revolution* (Sebastopol, California: O'Reilly & Associates, 1999) at 117.

[47] *Ibid.*

[48] Raymond, *supra* note 15.

[49] *Ibid.* at 88.

[50] *Ibid.* at 89.

[51] See P. Goldstien, *Copyright's Highway: From Gutenberg to the Celestial Jukebox* (New York: Hill and Wang, 1994). Under the incentive approach, the ultimate goal is efficient use of resources. Intellectual property law exists in order to provide adequate incentive for authors to produce works. Without copyright law, the thinking runs, authors have a diminished incentive to produce books, songs, and computer programs.

[52] See generally, T. Hardy, "Property and Copyright in Cyberspace" (1996) U. Chi. Legal F. 217. The property approach favours extending intellectual property protection much further than the incentive approach. The

property approach looks not only to provide an incentive to create works, but also to provide an incentive to exploit those works efficiently. The strongest economic justification for private property generally is that it internalizes costs and benefits.

[53] A comprehensive economic analysis of law supporting the Spartan approach is G. Lunney, Jr., "Reexamining Copyright's Incentive-Access Paradigm" (1996) 49 Vand. L. Rev. 483, and G. Lunney, Jr., "Trademark Monopolies" (1999) 48 Emory L.J. 367.

[54] See Raymond, *supra* note 15.

[55] See M. Radin, "Property and Personhood," (1982) 34 Stan. L. Rev. 957.

[56] W. Gordon, "A Property Right in Self-Expression: Equality and Individualism in the Natural Law of Intellectual Property," 102 Yale L.J. 1533 (1993). The author discusses Lockean and Hegelian notions of self-expression.

[57] McKeown, *Fox Canadian Law of Copyright and Industrial Designs* (Scarborough: Carswell, 2000) at 248. Moral rights have their origin in French civil law and are said to comprise four rights:

(1) the right of divulgation which allows the author to decide when to publish a work;

(2) the right of withdrawal or repentance which allows an author to withdraw a work from circulation to the public;

(3) the right of paternity which is the right to identify the author with his or her work; and

(4) the right of integrity which permits an author to maintain their work as they had expressed it, even after it is released to the public.

[58] *Ibid.*

[59] See S. McJohn, "The Paradoxes of Free Software" (2000) 9 Geo. Mason L. Rev. 25 at 9.

[60] *Ibid.*

[61] J. Hughes, "The Philosophy of Intellectual Property" (1998) 77 Geo. L.J. 287 at 299. For further discussion see A. Moore, "A Lockean Theory of Intellectual Property" (1997) 21 Hamline L. Rev. 65.

[62] A. Moore, *ibid.*

[63] L. Lessig, *The Future of Ideas: The Fate of the Commons* (Random House, 2001).

[64] *Ibid.*

[65] McJohn, *supra* note 59.

[66] Lunney, *supra* note 53 at 556–61.

[67] R. Stallman, "Reevaluating Copyright: The Public Must Prevail" (1996) 75 Or. L. Rev. 291.

[68] Lunney, *supra* note 53. It should be noted that Canadian trademark legislation is largely premised on consumer deception.

[69] See, for example, the opinions espoused by the Free Software Foundation available online <http://www.fsf.org/philosophy> (date accessed September 9, 2002).

[70] Open source development relies on a mass-market licensing model in order to control what is done with the source code. For more discussion on copyright law and open source licensing refer to subsection, "Copyright and Open Source Licensing".

[71] D. McGowan, "Legal Implications of Open-Source Software", (2001) U. Ill. L. Rev. 241. The author discusses the relevance of intellectual property laws in the development and maintenance of open source software.

[72] Inherited from the French system of Droit d'Auteur.

[73] D. Vaver, *Copyright Law* (Toronto: Irwin Law, 2000).

[74] *Copyright Act*, R.S.C., 1985, c. C.-42; amended by S.C. 1988, c. 65.

[75] Of particular importance is the case of *Apple Computer Inc. v. MacIntosh Computers Ltd.*, [1988] 1 F.C. 673. At issue was whether a computer program which originally existed in written text continued to be covered by copyright when it was embedded in a silicon chip designed to replicate the code. In essence, the Federal Court of Appeal laid down the general framework of copyright protection for computer programs leading to the 1988 amendments of the *Copyright Act*. See also, *Prism Hospital Software v. Hospital Medical Records Institute* (1994), 57 C.P.R. (3d) 129 (B.C.S.C.). The B.C. court took the American abstraction-filtration comparison test into consideration when determining if copyright infringement had occurred.

[76] The protection of "literary works" is found in s. 3 of the *Copyright Act*, R.S.C., 1985, c. C.-42; amended by S.C. 1988, c. 65.

[77] McKeown, *supra* note 57 at 164. The author refers to subsection 30.6 of the *Copyright Act*.

[78] *Ibid*, at 450.

[79] *Ibid.*

[80] *Ibid.*

[81] *Ibid.* at 457.

[82] Gomulkiewicz, *supra* note 14. The author argues that open source software would not be feasible without licensing regimes that deflect the risk of legal liability.

[83] C. DiBona *et al.*, *Open Sources: Voices from the Open Source Revolution* (Chris Dibona *et al.* eds., 1999) at 127.

[84] Keohane, "Mass Market Licensing" (2001) 652 PLI/P.

[85] N. Horne, "Open Source Software Licensing: Using Copyright Law to Encourage Free Use", (2001) 17 Ga. St. U. L. Rev. 863 at 871.

[86] Benlendorf, "Open Source as a Business Strategy", in *Open Sources: Voices from the Open Source Revolution*, *supra* note 46 at 167.

[87] For a comprehensive discussion on these types of licenses see Maher, *supra* note 24 at 639–642 and Horne, *supra* note 85 at 878.

[88] See The Open Source Definition at <http://www.opensource.org>.

[89] The label "open source" is protected through the licensing regime and through trademark law.

[90] The Open Source Definition, *supra* note 88. See also D. Ravicher, "Facilitating Collaborative Software Development: The Enforceability of Mass-Market Public Software License" (1999) 5 Va. J.L. & Tech. 11. Additionally, see P. Bobko, "Linux and General Public Licenses: Can Copyright Keep 'Open Source' Software Free?" (2000) 28 AIPLA Q. J. 81.

[91] The license may not restrict any party from either selling or giving away open source software. This license condition protects the freedom to choose to redistribute either "*gratis*" or for a fee.

[92] The license agreement must license the software in source code form. Such source code provided under the license must be in the preferred form a programmer would need to modify the program.

[93] The license agreement must grant the licensee the right to create modifications and derivative works. The License must explicitly permit distribution of software built from modified or derivative source code.

[94] Open source licensing requires that the author of a particular piece of code be acknowledged. This requirement is often satisfied by retaining the author's copyright notice on the code he or she creates as the code is passed on and modified further. A license may also require that derivative works be labelled with a different version number, or that their source code be distributed unmodified along with a mechanism that combines this code with modifications and derivatives when the software is actually compiled into binary or executable form for use by the computer. In addition, certain open source licenses prohibit the use of the name of the author of a given piece of code to endorse or promote products derived from that code.

[95] The license agreement must provide the software "as is", with no warranties either as to product performance or non-infringement of third-party intellectual property rights in order to shift the legal risk away from the code development.

[96] The rights attached to the software must apply to everyone to whom the software is redistributed. In other words, the licensee must agree to pass the open source license terms on to its licensees, and require those licensees to pass the terms on to all subsequent licensees.

[97] The license must not discriminate against any individual or group. In addition, the license must not restrict the use of the software in a particular field or endeavour. For example, the license may not restrict use of the software for business purposes or use in a controversial field of research, such as genetic engineering.

[98] The license must not place restrictions on other software distributed along with it.

[99] Ravicher, *supra* note 90.

[100] *Ibid* at para 75. The validity of the GPL license has neither been litigated in the United States nor in Canada. Ravicher discusses the ambiguities surrounding mass-market public software licenses such as the GPL. He further addresses the underlying public policy arguments for ensuring that such licenses are enforceable.

[101] See <http://www.xvid.org> (date accessed September 9, 2002).

[102] Shrinkwrap and clickwrap agreements are seen as unilateral or adhesive contracts; one party dictates the terms to be contracted to without the opportunity of the other party to negotiate such terms. Typically, clickwrap contracts utilize pop-up boxes in which the terms and conditions may be scrolled through in succession. It has been contended that because only a portion of the agreement is on the screen at any one time, the remainder are equivalent to the fine print typically found in printed standard forms. Shrinkwrap licenses may take several forms. A typical example is where the licensing terms are found printed on the outside of a box containing computer software. Sometimes the boxes are further packaged by a protective layer of plastic. In theory, the consumer will have read the licensing terms before tearing the plastic or opening the box. By opening the plastic or the box, consumers bind themselves to the terms of the license.

[103] P. Bobko, *supra* note 90 at 104. See also D. Ravicher, *supra* note 90.

[104] *Ibid.*

[105] *Rudder v. Microsoft*, [1999] O.J. No. 3778 (Ont. Sup. Ct.). The court held that the clickwrap agreement in dispute was analogous to "fine print" in a written contract. See also: *Kanitz et al. v. Rogers Cable Inc.* 58, O.R. (3d) 299, [2002] O.J. No. 665.

[106] *Uniform Electronic Commerce Act* available online at <http://www.ulcc.ca/en/us/index.cfm?sec=1&sub=1u1> (date accessed September 9, 2002).

[107] *ProCD v. Zeidenberg*, 86 F. 3d 1447 (7th Cir. 1996).

[108] See G. Founds, "Shrinkwrap and Clickwrap Agreements: 2B or not 2B?" (1999) 52 Fed. Comm. L.J. 99. See also, D. Karjala, "Federal Preemption of Shrinkwrap and On-Line Licenses", 22 U. Dayton L. Rev. 511.

[109] For further discussion on the validity of shrinkwrap licenses, see Keohane, *supra* note 84 at 437. The author contrasts the decision of *Step-Saver Data Sys. v. Wyse Tech.*, 939 F. 2d 91 (3d Cir. 1991) with that of *ProCD v. Zeidenberg*, 86 F. 3d 1447 (7th Cir. 1996).

[110] G. Takach, *Computer Law* (Toronto: Irwin Law, 1998) at 283.

[111] GPL license available at <http://www.gnu.org/copyleft/> (date accessed September 9, 2002).

[112] McJohn, *supra* note 59.

[113] See generally P. Goldstein, *Copyright, Patent and Trademark* (Westbury: Foundation Press, 1992).

[114] B. Perens, "The Open Source Definition" at <http://www.opensource.org> (date accessed September 9, 2002).

[115] *Ibid.*

[116] Goldstein, *supra* note 113.

[117] Press announcement of "OSI Certified" issued on June 16, 1999 at <http://opensource.org> (date accessed September 9, 2002).

[118] Vaver, *supra* note 73.

[119] McJohn, *supra* note 59.

[120] Goldstein, *supra* note 113.

[121] See R. Stallman, "The GNU Operating System and Free Software Movement", in Open Sources: Voices from the Open Source Revolution, *supra* note 46 at 67. See also, McJohn, *supra* note 59 at 11.

[122] McJohn, *supra* note 59.

[123] S. Garfinkel, "Patently Absurd", on <http://www.wired.com> (search site).

[124] See generally comments from <http://www.slashdot.com> (date accessed September 9, 2002).

[125] McJohn, *supra* note 59 at 12.

[126] The recent call for the adoption of a royalty-bearing patents for Web standards in the form of a new RAND license has been met with much criticism from software developers. Nonetheless, the Patent Policy Working Group (PPWG) is considering a royalty based model for www standards. See A. Orlowski, "WWW Royalties Considered Harmful" at <http://www.theregister.co.uk/content/6/22561.html> (date accessed September 9, 2002).

[127] *Ibid.*

[128] See S. Shulman, "Software Patents Tangle the Web", in Technology Review at <http://www.techreview.com> (date accessed September 9, 2002).

[129] Goldstein, *supra* note 113.

[130] Although software patenting has been around for several years, it is still relatively recent when compared to other patented items, such as pharmaceuticals.

[131] M. Haynes, "Commentary: Black Holes of Innovation in the Software Arts" (1999) 14 Berkeley Tech. L.J. 567. Haynes discusses the trend in software patenting noting, in particular, the stifling to innovation by antiquated reverse-engineering provisions in the American system. The author discusses the role of open source and the impact that patents may have on its development.

[132] G. Ahorian, Internet Patent News Service at <http://www.bustpatents.com> (date accessed September 9, 2002).

[133] E. Raymond, "Open Source Software A (New?) Development Methodology" at <http://www.opensource.org/halloween/halloween1.php> (date accessed September 9, 2002). The author presents an annotated memorandum originating with discussion on Microsoft.

[134] See N. Petreley, "Linux and the Monopoly Game", at <http://www.linuxworld.com/linuxworld/lw-1999-01/lw-01-penguin.html> (date accessed September 9, 2002).

[135] See P. Rooney, "Linux is Top Threat to Windows", at <http://techweb.com>. See also S. Shankland, "Linux Shipments up 212 Percent" at <http://news.com.com/2100-1001-219214.html?legacy=cnet>; B. Sullivan, "Linus Torvalds — Microsoft Killer?" At <http://www.msnbc.com> (sites accessed September 11, 2002).

[136] See "Microsoft Executive Says Linux Threatens Innovation", at <http://www.cnetinvestor.com> (date accessed September 11, 2002).

[137] See Raymond, *supra* note 15.

[138] Other authors argue the converse, that closed proprietary models are more beneficial to society. See, for example, M. Strasser, "A New Paradigm in Intellectual Property Law? The Case Against Open Sources" (2001) Stan. Tech. L. Rev. 4.

[139] See E. Raymond, *supra* note 15.

[140] *Ibid.*

[141] See M. Lemley and D. McGowan, "Legal Implications of Network Economic Effects" (1998) 86 Cal. L. Rev. 479 at 528-30. See also M. Lemley and D. McGowan, "Could Java Change Everything? The Competitive Proprietary of a Proprietary Standard" (1998) 43 Antitrust Bull 715.

[142] McKeown, *supra* note 57 at 60.

[143] *Ibid.* at 527–30.

[144] *Ibid.*

[145] In its more generic sense, reverse engineering refers to taking an object apart to see how it works, then making changes to enhance the object. Reverse engineering of software is a process where a computer programmer will send object code (the string of 1s and 0s) to the source code of a software program in order to retrieve the source code. Access to the source code allows a computer programmer to study how a software program operates. The computer programmer may then make adjustments to the source code to fix any problems or to enhance the program.

[146] M. Lemley and D. McGowan, *supra* note 141. The recent release of Windows XP is an example of the controversy surrounding the frequency of Microsoft software upgrades; see M. Tiemann, "Windows XP: Extra Proprietary" at <http://www.redhat.com/about/opinions/xp.html> (date accessed September 9, 2002).

[147] See <http://www.winehq.com> for details on this project (date accessed September 9, 2002).

[148] See Raymond, *supra* note 15 at 129–161. A market positioner or loss leader uses open source to create or maintain a market position for proprietary software that generates a direct revenue stream. Open source client software can enable sales of server software, or generate advertising revenue on an Internet portal. The author uses the example of the Mozilla browser used by Netscape Communications.

[149] *Ibid.* Widget frosting is a model for hardware manufacturers because they view software as overhead, rather than a profit center. Market forces have compelled hardware manufacturers to write and maintain software, such as device drivers. Raymond notes that if they had used open source software, their maintenance and writing costs could have been 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".

[150] *Ibid.* "Giving away the recipe and opening a restaurant" means selling value added services by assembling and testing a running operating system that is warranted to be merchantable and plug-compatible with operating systems carrying the same brand. Companies such as Red Hat generate revenue by doing just this and by selling services to install and provide support service contracts.

[151] *Ibid.* Accessorizing means selling accessories for open source software. For example, O'Reilly Associates publish many excellent reference volumes on open source software.

[152] *Ibid.* "Free 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. Thus, the software developer has a short term monopoly in which he or she is the sole profit recipient. Upon expiration of the term, others have the right to alter the software and to create derivative products.

[153] *Ibid.* The notion of brand equity, as Raymond notes, is a selling point to a much greater extent than the underlying technology. 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. OSI Certification is hoped to strengthen industry standards and to promote brand recognition.

[154] *Ibid.* "Freeing the software, selling the content" means giving away software that is only a means to something else. As Raymond observes, "the value is neither in the client software nor the server, but in providing objectively reliable information". He further notes that America Online should move its client software to open source for this very reason.

[155] L. Lessig, *Code and Other Laws of Cyberspace* (Basic Books, 1999).

[156] *Ibid.*

[157] L. Lessig, "The Limits in Open Code: Regulatory Standards and the Future of the Net" (1999) 14 Berkley Tech. L.J. 459 at 487.

[158] G. Lunney, Jr., "The Death of Copyright: Digital Technology, Private Copying, ad the Digital Millennium Copyright Act," (2001) 87 Va. L.Rev. 813. The author suggests that the enactment of the DMCA would have the effect of displacing public interest in copyright with an imbalanced regime concerned primarily with private interests.

[159] B. Fitzgerald, "Software as Discourse: The Power of Intellectual Property in Digital Architecture" (1999) 18 Cardozo Arts & Ent. L.J. 337 at 338.

[160] *Ibid.*

[161] J. Cohen, "Taking Stock: The Law & Economics of Intellectual Property Rights", (2001) 53 Vand. L. Rev. 1799. The author argues that a new approach to copyright should be adopted. She suggests that approaches should examine how to optimize the creative process. Optimizing the creative process must involve the formation open spaces, "zones of unpredictability within an around the predictable contours of rights and rules". Such open spaces may include, but are not limited to, support for fair use, reverse engineering and open source.

[162] For further discussion see N. Weinstock-Netanel, "Copyright and a Democratic Civil Society" (1999) 1-6 Yale L.J. 283.

[163] *Digital Millennium Copyright Act*, 17 U.S.C. [hereinafter DMCA].

[164] Canada, along with the United States, Europe, Japan and Australia are signatories to the *World Intellectual Property Organization Copyright Treaty* which requires that states promulgate effective anti-circumvention measures. Canada is expected to ratify this treaty and to, subsequently, implement anti-circumvention measures.

[165] See Y. Benkler, "Free as the Air to Common Use: First Amendment Constraints on Enclosure of the Public Domain" (1999) 74 N.Y.U.L. Rev. 354.

[166] D. Nimmer, "A Riff on Fair Use in the Digital Millennium Copyright Act" (2000) 138 U.Pa.L.Rev. 673.

[167] J. Ginsburg, "Copyright Legislation for the 'Digital Millennium'" (1999) 23 CLMVJLA 137.

[168] Y. Benkler and L. Lessig of Amici Curiae in Support of Appellant (Jan. 26, 2001); *Universal City Studios v. Shawn C. Reimerdes*,00 Civ. 0277 (LAK), S.D.N.Y., Aug. 17, 2000.

[169] L. Lessig, *supra* note 155.

[170] *Ibid.* at 100–108.

[171] *Ibid.* at 113.

[172] "Anarchy" may not be an accurate picture of open source software development given that many terms are imposed on developers through licenses. I am not the first person to describe the movement as a form of anarchy. Others such as Lessig and McJohn imply the anarchist nature of open source development.

[173] Lessig, *supra* note 155.

[174] The *Napster* and *Reimerdes* debacles are illustrative of the competing tensions between market players who wish to maintain control over the media industry in the digital era with the users of these technological products who wish to see a new balance struck. For references to these cases refer to *A & M Record v. Napster*, 239 F. 3d 1003, C.A. 9 (Cal.) 2001; and *Universal City Studios v. Shawn C. Reimerdes*, 00 Civ. 0277 (LAK), S.D.N.Y., 2000.

[175] S. Handa, "Reverse Engineering Computer Programs Under Canadian Copyright Law." (1995) 40 McGill L.J. 621.

[176] S. McJohn, *supra* note 59.

[177] Many grassroots groups have sprung up vocalizing similar opinions. In an online article available at <http://www.linuxplanet.com/linuxplanet/reports/4325/1/>, D. LeBlanc provides a list of active organizations including: Free Software in Government <http://www.memeshadow.net/npiwiki/index.php/Free%20Software%20In%20Government>, CanOpener which is the Canadian Open Source Education and Research Organization <http://www.canopener.ca>, and Canadian Linux User's Exchange <http://www.linux.ca> and Flora <http://www.flora.ca>. All accessed September 9, 2002.

[178] *Copyright Act*, R.S.C. 1985, c. C-42, ss. 29, 29.1 and 29.2 as amended.

[179] *Ibid.* at ss. 29.3, 29.4, 29.5, 29.6, 29.7, 29.8, 29.9 and 30.

[180] *Ibid.* at ss. 30.1, 30.2, 30.21, 30.3, 30.4 and 30.5.

[181] *Ibid.* at s. 30.6.

[182] *Ibid.* at s. 30.7.

[183] *Ibid.* at s. 30.8 and 30.9.

[184] *Ibid.* at s. 80 as amended.

[185] In an article written by Michelle French, Syd Weidman, the president of the Prairie Linux User Group (PLUG), commented that removing or rendering ineffective the reverse engineering defense, "could spell the end for free [open] software". This is perhaps somewhat exaggerated. Not all open source projects require the reverse engineering of proprietary software programs. Many are initiated at the ground level where the code is written for the first time, then modified as the project progresses.

[186] The study is available at <http://www.linux-quebec.org/> (date accessed September 9, 2002). Laval University will likewise produce a study on open source. See D. LeBlanc, "Linux in Canada: Are We Going Open Source Yet?" available at <http://linuxplanet.com/linuxplanet/reports/4325/2>.

[187] <http://www.linuxsymposium.org> (accessed September 9, 2002).

[188] <http://bosc.open-bio.org> (accessed September 9, 2002).

[189] LeBlanc, *supra* note 186.

[190] See <http://www.internetnews.com/bus-news/> (accessed September 9, 2002).

[191] Available at <http://www.naci.org.za> (accessed September 9, 2002).

[192] *Ibid.*

[193] *Ibid.*

[194] A draft of the *Security Systems Standards and Certification Act* is provided by Declan McCullagh at <http://www.politechbot.com/docs/hollings.090701.html> (accessed September 9, 2002).

[195] See articles by T. Gasperson and D. McCullagh at <http://www.linuxtoday.com/news_story>. It should be noted that the SSCA is a work in progress. Given the growing importance of Linux in the market, it is unlikely that this is merely aspect of the Bill was overlooked, and that the final draft of the Bill would not prohibit the Linux kernel.