THE GOVERNANCE OF CODE: CODE AS GOVERNANCE

Serena Syme[1]

Masters of Public Policy
Kennedy School of Government
Harvard University
Cambridge, MA 02138
scsyme@hotmail.com
617-596-4738

L. Jean Camp[2]

Assistant Professor
Kennedy School of Government
Harvard University
Cambridge, MA 02138
jean_camp@harvard.edu
617-496-6331
www.ljean.net

The governance of a network society is tightly bound by the type of license generated by the creation of information property. The establishment of a market involves the development of a bundle of rights that both create property and define the rules under which property-based transactions might occur. The fundamental thesis of this work is that the creation of property through licensing offers different views of the governance of the network society.

Thus this article defines the network society by considering the various forms of governance currently applied to code, namely: open code licensing, public domain code, proprietary licenses, and the Uniform Computer Information Transactions Act (UCITA). The open code licenses addressed here are the GNU Public License, the BSD license, the artistic license, and the Mozilla license.

We posit that the licenses are alternative viewpoints (or battles) over the nature of the network society, and that each has its own hazards. We describe the concepts of openness: free redistribution, source availability, derivations, integrity, non-discrimination, non-specificity, and non-contamination. We examine how each license meets or conflicts with these conditions.

We conclude that each of these dimensions has a parallel in the dimension of governance. Within our conclusions we identify how the concept of code as law, first described by Stallman and popularized by Lessig, fails when the particulars of open code are examined. However, we explore the ways that licenses together with code provide a governance framework, and how different licenses combined with code provide a range of visions for governance of the information society. We go on to consider the fundamentally different governance model outlined by UCITA, and comment on the philosophical implications and hazards of such a framework for the world of code.

INTRODUCTION

In this work we focus on code, rather than on information goods as a class. Doing so provides for us a framework in which to explore the implications of different mechanisms of governance as applied to the code underlying the network society. We consider the use of open code and its associated licensing frameworks, and move on to licensing as outlined by the proposed Uniform Computer Information Transactions Act (UCITA). We describe the differences between code as distinguished by the governance mechanism.

Part 1 begins by defining open code (or free) software, and contrasting it with public domain software.[3] It then moves to consider some of the characteristics of open code licenses, before examining four such licenses in some detail. Part 2 continues with a consideration of UCITA, its effects, and its likely impact on consumers and the open code software movement.

In Part 3 we consider the governance models offered by the licenses and by UCITA. We close by arguing that different visions of society are implicit in different licensing mechanisms for code, and describe the philosophical and governance implications of open code licensing and the UCITA model.

PART 1: SOFTWARE LICENSES AND OPEN CODE

What is open code? A tremendous amount of attention has been focused on open code per se, with little corresponding discussion regarding the modes of open code, their differences, and the substance of those differences.

Open Code Software vs. Public Domain Software

Open source or free software is not necessarily free in terms of price - the concepts refer to the lack of proprietary restrictions on ideas underlying the code. "Free software" requires the freedom to use, modify, and distribute the software. "Open" means open to examination and alteration.

It is often assumed that public domain software is free software, but both open code and free software proponents disagree with this view. Public domain software has been released to the public without any copyright (or more correctly, with an express rejection of the author's copyright). This enables users to do whatever they like with that code, including re-copyrighting it as their own work. The result is that users may then convert the program into proprietary software (with major or minor modifications), and redistribute it in this form. This action will strip away the freedom originally attached to the program.

Free software proponents, such as the original advocate of free software, Richard Stallman, argue on this basis that public domain software does less service to the community than free software. Free software is distributed with a license that, in simple terms, prevents it from being made proprietary in the future. As a result, it is arguably "more free" than public domain software. Free software is more free precisely because of the controls on its use -- that is, the governance mechanism of free software guarantees continued freedom.

Free Software Licenses

The Free Software Foundation (FSF) assesses software licenses to determine whether or not they are free, and whether or not they provide a true copyleft (Free Software Foundation, 2000a). "Copyleft" was designed by Stallman, the founder of the Free Software Foundation, as a means to keep software free rather than private. According to him:

"The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program and distribute modified versions - but not permission to add restrictions of their own. Thus the crucial freedoms that define 'free software' are guaranteed to everyone who has a copy; they become inalienable rights." (Stallman, 1999, p.59)

Licenses can be free but not provide a true copyleft (one example is the BSD License). The FSF states that free, but non-copyleft, software is issued by the author with permission to redistribute and modify, and also to add additional restrictions to it (Free Software Foundation, 2000b). A license cannot provide copyleft if it is not free. According to the FSF, the GPL is the only license that is both free and contains a strong copyleft provision.

The criteria used by the FSF to judge whether a license is free or a copyleft are distinct from those contained in the Open Source Definition (below). There are numerous examples of software that is open source but not free. For these reasons we include free software and open source as distinct categories.

The Open Source Definition

Bruce Perens of the Open Source Initiative (OSI) describes the Open Source Definition he prepared as a "bill of rights for the computer user" (Perens, 1999, p.171). A license must fall within this Definition to be certified as "open source " by OSI.

According to the Definition, the distribution terms of an open source software product must satisfy the following conditions:

- *Free redistribution:* The license must not restrict the sale or disposal of the software, or require any fee in connection with such a sale (however, an individual distributor may charge a fee if he or she chooses).
- *Source code:* The software must include or allow easy and free access to its source code, and this code must be in the preferred form for programmer modifications.
- *Derivations:* The license must expressly allow the software to be modified, and must allow any modifications or derived works to be distributed under the same license.
- *Integrity:* The license may require any modifications to the source code to be documented and/or included in patch files. This is more of a concession from the derivation condition than a condition in its own right - it allows authors limited control over derivations in order to protect the integrity of their code.
- *No discrimination:* The license must not discriminate against persons or groups, or against use in a specific field of endeavor.[4]

- *License distribution:* The rights must apply to all recipients of the software, without the need for execution of an additional license.
- *Non-specificity:* The rights must not depend on the software being part of a specific distribution.
- *Non-contamination:* The license must not impose restrictions on other software distributed with the licensed software.

Notice that Java could not have originally been considered open code, because there were limitations on the use of Java 1.1 in safety-critical systems. Thus the original Java code, though arguably superior in terms of governance, would not be open.

This paper assesses the GNU General Public License, the BSD License, the Netscape/Mozilla Public License and the Artistic License against these criteria, and also compares and contrasts the protections they offer.

GNU General Public License

In many ways, the GNU General Public License (GPL) is the archetypal free software license, so one would think it would necessarily comply with the conditions for an open source license set out in the Open Source Definition.

Free redistribution: Article 1 of the GPL allows for free copying and distribution of the program, provided that both copyright and the license are maintained on any new copies. It allows for a fee to be charged for this service, but does not require that a fee be charged.

Source code: GPL Article 3 requires that machine-readable source code be made available with the licensed software. Source code is defined as "the preferred form of the work for making modifications to it".

Derivations: Article 2 allows the software to be modified, and requires that modified works be distributed under the GPL. The GPL is "viral" in that any works that include sections of code derived from GPL licensed software must themselves be GPL licensed. This creates a strong "copyleft" - the free software distributed under the GPL cannot legally pass out of the free software domain. Therefore the GPL satisfies this criterion. However, this also means that GPL-licensed code cannot be mixed with non-free software, which is a significant restriction on its use. The GPL is the only one of the major free software licenses that has this effect.

Integrity: Article 2 expressly allows the distribution of modified code, and requires that the modified files carry prominent notices indicating that they have been changed.

No discrimination: Article 8 of the GPL does allow the license to restrict distribution in certain countries, if due to the intellectual property system in those countries, distribution there would infringe the GPL. Although this may be considered to be discrimination, it is better viewed as a rejection by those potential recipients of the terms of the GPL.

License distribution: Rights under the GPL apply to all recipients of software covered by that license. There is no requirement that an additional license be executed.

Non-specificity: The GPL does not allow rights to depend on the software being part of a specific distribution - GPL protection is ongoing and viral.

Non-contamination: The GPL does impose restrictions on software that forms part of a modified work together with some portion of the licensed software. But it will not apply to identifiable sections that "are not derived from the Program and can be reasonably considered independent and separate works in themselves", and neither will it apply to separate works that are aggregated with licensed software on a storage or distribution medium (Article 2). Consequently, the GPL arguably satisfies this requirement. However, there is some debate on this point within the open source community and among open source scholars.

It follows that the GPL complies with the conditions set out in the definition, and that any software distributed under the GPL would be considered to be open source software. Modifications to and developments based on GPL software remain in the free software domain, and consequently can be used to improve the original product.

The LGPL, also called the lessor GPL, is similar to the GPL and was designed by GNU to cover software libraries (thus the preceding L). For this reason, it does allow protected code to be incorporated into proprietary

software. The creation of the LGPL was necessitated by the use of proprietary elements, for example in device drivers and library expansions for proprietary systems.

BSD License

The BSD License is a non-copyleft free software license used by the various Berkeley Software Distribution (BSD) projects and by Apache. Behlendorf characterizes the license as follows: "Here's this code, do what you like with it, we don't care, just give us credit if you try and sell it" (Behlendorf, 1999).

The most common criticism of this license relates to its requirement that attribution be made to Berkeley in all uses or advertisements of the software. The FSF outlines the practical problems caused by this clause, namely that developers applying the BSD License to their own software have tended to copy this clause and change it to refer to their own company or university. In a large collaborative software project, this can result in a requirement that many lines of advertising be used to acknowledge all these contributors. The FSF notes that the NetBSD distribution requires at least 75 of these sentences (Free Software Foundation, 2000c). This characteristic makes the license incompatible with the GNU GPL, because software originally released under the BSD but later forming part of a GPL-protected project does not receive the protection it needs from the GPL, and so the advertising clause in the original license must still be observed.

The Modified BSD License addresses this concern (by deleting this requirement and eliminating references to Berkeley) while retaining all of the other characteristics of the BSD license. The Modified BSD License is compatible with the GNU GPL, but is also a non-copyleft license. The Modified BSD License is what allowed BSD, rather than GNU-Linux, to be the basis of Macintosh Operating System X.

The license contains no incentive to contribute modified code back to the original project.

Free redistribution: The BSD license does not restrict the sale or disposal of the software, except to the extent that it requires that certain attribution and legal requirements (specifically, inclusion of a copyright notice and a disclaimer) be met.

Source code: The BSD license does not include any requirement that source code be provided with the software, although it allows such a distribution. In this sense, the license is not a copyleft - it fails to guarantee essential free software rights to subsequent recipients of the software.

Derivations: Modifications are allowed under the BSD License, and there is no provision as to how those modifications should be treated for licensing purposes. Consequently, they could be (but are not necessarily) protected by the BSD License, which therefore satisfies this criterion.

Integrity: The BSD License expressly allows the distribution of software built from modified source code. However, there is no requirement that modifications be documented.

No discrimination: The BSD License does not discriminate against any potential users.

License distribution: There is no requirement under the BSD License that recipients execute an additional license. However, the license is not "viral", and as a result "second generation" recipients of the software may not enjoy free software rights.

Non-specificity: The rights attaching to BSD-licensed software do not depend on the software being part of a specific distribution.

Non-contamination: The BSD License is permissive and does not impose restrictions on any other software.

The Apache License is very similar to the original BSD License and includes an advertising clause and also a similar clause requiring attribution to Apache in any distribution. The history of BSD suggests that both of these provisions are likely to give rise to practical problems, and make the Apache License incompatible with the GNU GPL.

Mozilla Public License

The Mozilla Public License (MozPL) was developed to cover the public release of the Netscape Navigator code (dubbed "Mozilla") in 1998. Other free software licenses could not be used for this purpose because many of the third party components contained in the Mozilla code were covered by private licensing agreements. The Mozilla license needed to be compatible with those private licenses and also allow commercial developers to

contribute code to subsequent versions of Mozilla (Hammerly et al, 1999).

The first version of the license - the Netscape Public License (NPL) -- was publicly "beta-tested". It was posted on a public newsgroup and amended in accordance with the comments received. These comments resulted in the development of a second license intended to work with the NPL, namely the MozPL. The NPL is identical to the MozPL except that it confers additional rights on Netscape.

The NPL and MozPL both allow modifications to protected code to be "taken private" and not kept in the open software domain, but they are flexible in allowing the combining of protected code and proprietary code. Both the NPL and the MozPL are free software licenses, but they do not provide a strong copyleft, no doubt in part because of the desire of Netscape (now AOL) to incorporate any technology integrated with Mozilla. Netscape saw this as necessary given that the bulk of contributions to Mozilla were initially made by Netscape employees. As a result of this need by Netscape/AOL to protect its own perceived investment, these licenses are not compatible with the GNU GPL. The licenses are fairly opaque in their drafting and would benefit from a plain English rewrite. In the following paragraphs we have attempted to extricate the pertinent details from the licenses.

Free redistribution: Under Article 2.1 of the licenses, Netscape confers on a person in compliance with them a "world-wide, royalty-free, non-exclusive license" to use, distribute and sell (among other things) the code. Other contributors to the code also confer equivalent rights on such a user (Article 2.2). A person may only distribute an executable form of the code if they do so in compliance with Articles 3.1-3.5, which include requirements regarding notices and the availability of source code.

Source code: Article 3.2 requires that any modifications must be made available in source code form either with the executable version, or by electronic distribution (such as by email). The definition of source code specifies that it is the preferred form of the code for making modifications to it. A distribution of the source code must be accompanied by a copy of the license (Article 3.1). Article 3.6 indicates that "You" may choose to distribute the code under another license, but that license must be compatible with MozPL/NPL (all the rights and obligations under this license must be conferred on subsequent users), and the distribution itself must be in compliance with this license. In practice, this means that any such license will likely be very similar to MozPL or NPL, but it may confer additional rights on users.

Derivations: Article 2 allows the software to be modified, and Article 3 ensures that any such modifications are covered by the license. Article 3 provides that any modifications that "You" make to the code are subject to Article 2.2, meaning that you are then required to grant licenses to others to use that modified code.

Note that no patent license is granted under Article 2 in the case of infringements caused by the modification of the code or the incorporation of the code with other software. This allows the original developer (Netscape) to retain the ability to enforce its patent rights against a modifier of the code, or against a user of the modified code. This appears to be a fairly far-reaching exception, although in reality Netscape is likely to restrict any prosecutions to commercial uses of the code.

Integrity: the license does allow the distribution of modified code, and Article 3.3 requires any modifier to include with the code a file listing all changes made and the dates of those changes, and also a statement of attribution regarding the original developer of the code (usually Netscape).

No discrimination: the license does not discriminate against persons or groups, or against use in a specific field of endeavor.

License distribution: the license applies to all recipients - there is no requirement that they execute an additional license. Article 2 of the license automatically confers certain rights on recipients, and Article 3.6 ensures that modifiers or developers of the code cannot require recipients to execute additional documents before enjoying those rights.

Non-specificity: the rights conferred by Article 2 apply to the original code and to any modifications to that code or larger works in which the code is incorporated. The rights are not limited to specific distributions of code.

Non-contamination: the license only applies to the original code (described as such in Exhibit A) and modifications to that code (narrowly defined as additions to or deletions from the substance or structure of the code or previous modifications). It does not impose restrictions on other software distributed with the licensed software.

Article 3.5 requires fairly extensive attribution of authorship in the code. This is likely to give rise to the same practical problems described in relation to the BSD License. Behlendorf identifies a further flaw, which is that the patent waiver in Article 2.2 would require a contributor to waive its patent claims over the entire Mozilla code, not just its contribution. This is likely to cause headaches for large companies, which may have patent rights to other parts of the code (Behlendorf, 1999).

Article 10 of this license addresses the particular legal restrictions applying to government users - such a provision is not included in the other licenses.

The NPL differs from the MozPL in the following respects. First, it protects the Netscape name and logo. Further, it allows Netscape to include third party code in the software without releasing the source code if its pre-existing license agreements with the third party prohibit such a release. But the sticking point for many developers is that the license allows Netscape to incorporate the code covered by the license (including modifications contributed by third parties) in any of its own proprietary products for a period of 2 years (MozPL Amendment V2).

Furthermore, Mozilla-based code (also including modifications contributed by third parties) can be later distributed under its Netscape trademark on different terms from those contained in the NPL, and presumably on a proprietary basis (MozPL Amendment V3). There is no time limit on this activity. These constraints have been identified as a major obstacle for the open code community. A contributor may find him or herself unable to use his or her code as integrated under Mozilla if the code is later released under the proprietary Netscape license rather than being integrated into Mozilla. In effect, contributions to Mozilla may become contributions to AOL.

Artistic License

The Artistic License was developed for a language, not to protect a product produced in a particular language. It was created by Larry Wall for the Perl language, and has since been heavily criticized but widely adopted. Perens states that "it is, in my opinion, a sloppily-worded license, in that it makes requirements and then gives you loopholes that make it easy to bypass the requirements" (Perens, 1999, p.183). The FSF comments that "We cannot say that this is a free software license because it is too vague; some passages are too clever for their own good, and their meaning is not clear" (Free Software Foundation, 2000d).

The Artistic License is considered to be an open software license, but not a copyleft.

Free redistribution: Article 1 allows a recipient to make and give away copies of the standard software package without restriction, as long as copyright notices and disclaimers are also copied and distributed with the code. But note that this may not apply to the package if a third party has modified it in a non-standard way, so the reach of the license may be quite limited in practice.

A distributor may not demand a fee for the software, but may charge reasonable copying fees and any support fees it decides (Article 5). However (and this is a significant loophole), a fee may be charged for the software if it is aggregated as part of a larger software distribution (Article 5). The free redistribution condition of the Open Software Definition was worded specifically to include this "aggregation" scenario, even though it is not strictly within the spirit of free software. Note also that if the protected code is fully embedded in another distribution, the rights no longer appear to apply (Article 8).

Source code: Article 4 ensures that the source code is distributed with the software, or made available to the recipient.

Derivations: The license allows for modification of the code, provided that the modification is fully documented, and steps are taken to ensure that those modifications are available to others, as appropriate (Article 3). It does not specify whether modified code can be distributed under the same license, but there is no prohibition against this (and given that the license is intended to protect Perl projects, its applicability to modified versions may be assumed).

Integrity: As noted above, modifications must be fully documented.

No discrimination: The Artistic License does not discriminate against any category of potential recipients.

License distribution: The License does not require execution of any additional license.

Non-specificity: The License specifically allows for the protected code to be aggregated with other code (Articles 5 and 8).

Non-contamination: The license is not viral; it makes no attempt to restrict the rights attaching to other software with which it is distributed.

Comparisons of Open Code

In the previous section we have described the basis for open code and free software, and have examined how distinct licenses fit and conflict with the Open Source Definition. In the next section we will describe the canonical closed license, as outlined by UCITA. Then we will discuss how the elements of the Open Source Definition map onto concepts of governance.

Before moving forward into a discussion of UCITA, a summary of the distinctions between the various open source licenses is in order. The following summary shows how the various licenses differ in terms of the authors' understanding of concepts of openness:

	Public Domain	GPL	LGPL	Mozilla	BSD	Artistic
Free redistribution	•	•	•	•	•	Limited
Source availability	•	•	•	•	•	•
Derivations	Yes, but not viral	Yes, viral	Yes, viral	Yes, somewhat viral	Yes, but not viral	Yes, but not viral
Integrity		•	•	•		•
No discrimination	•	•	•	•	•	•
Non-specificity	•	•	•	•	•	•
Non-contamination	•	?	•	•	•	•

Table 1: Requirements of the various open code and free software licenses.

PART 2 - THE UNIFORM COMPUTER INFORMATION TRANSACTIONS ACT

Bills to codify the Uniform Computer Information Transactions Act (UCITA) have been passed in two States and are pending in several others. Once passed, UCITA will have a significant effect on software licensing frameworks in participating States. Thus UCITA has the potential to become the default licensing framework for all software. Despite the far-reaching effects of UCITA, it is widely seen as a merely technical bill in governance circles. This paper aims to counter that mistaken perspective.

UCITA is addressed in a different section than the plethora of licenses above because it is fundamentally different. All of the previous licenses view source code, to one degree or another, as conversational creations – creative speech acts capable of modification and development. UCITA views code as a service to be licensed for a specific purpose and duration.

Background to UCITA

Under the United States' Uniform Commercial Code (UCC), "shrink-wrap licenses" (the one size fits all licenses included with packaged software or appearing on-screen before software can be used) are generally considered to be invalid. The reasoning is that the consumer has already purchased the product; consequently, it is not a true contractual exchange, so it is too late for software companies to be imposing conditions at that stage.

As a result, the exclusions of warranty and limitations of liability customarily included in those licenses are likely to have no effect. Naturally, proprietary software companies were unsatisfied with this position. Such companies banded together to form the Business Software Alliance (BSA), which sought to influence the National Conference of Commissioners on Uniform State Laws' (NCCUSL's) development of a uniform law to address the situation - the proposed Uniform Computer Information Transactions Act (UCITA). BSA's

membership includes most of the large U.S. software producers, and the promotion of UCITA is a prominent element of its policy platform.[5]

The Core of UCITA

UCITA has been prepared and approved by the NCCUSL, which prepares uniform laws to boost legal consistency from State to State. It has already been passed as a law in Virginia and Maryland.[6] The intent of UCITA is to establish a default position in which software developers and distributors are liable for flaws in a program, but also allow the use of shrink-wrap licenses to eliminate this liability. UCITA also imposes a number of restrictions on consumers' use of software products.

The NCCUSL describes the purpose of the law as follows:

"The Uniform Computer Information Transactions Act represents the first comprehensive uniform computer information licensing law. This act uses the accepted and familiar principles of contract law, setting the rules for creating electronic contracts and the use of electronic signatures for contract adoption-thereby making computer information transactions as well-grounded in the law as traditional transactions." (NCCUSL, 2000)

UCITA establishes a new contractual framework to cover computer information. The NCCUSL states that "most of the rules in UCITA are the traditional and familiar rules of contract from the law of sales and from the common law, but adapted to the special nature of computer information licensing contracts" (NCCUSL, 2000). The rules are "default", in the sense that the parties to a contract can agree that certain UCITA rules will not apply. Further, in most cases the parties' contract will prevail to the extent of any inconsistency between it and UCITA.

The more controversial effects of UCITA can be summarized in the following eleven sub-sections.

1. Coverage

In a State where UCITA applies, it will cover contracts entered into after its commencement date if their primary purpose is to create, modify, transfer or license "computer information" (section 103).[7] UCITA does not apply to goods unless the main purpose of the transaction is to obtain software embedded in those goods. The drafters of UCITA have confirmed that it is not intended to cover toasters, televisions, cars or other consumer goods in which embedded software is a minor part of the transaction.[8]

In practice, UCITA will cover contracts for software, on-line databases, Internet applications, HTML source code, computerized music/MP3s and a wide range of other types of digital information. UCITA will also automatically apply to storage devices that exist primarily to hold computer information, for example personal digital assistants. Other products that contain computer information may be brought within the UCITA framework by express reference in a contract, which is known as "opt-in".

UCITA will not automatically apply to books, newspapers, magazines, television or movies. However, the opt-in provisions allow a vendor or publisher of information to bring any information-based transaction within the Act. Section 103(e) provides that "if the subject matter of a transaction includes information, parties may agree that this [Act], including contract formation rules, governs the transaction in whole or in part or that other law governs the transaction and this [Act] does not apply". In this context, information is defined to mean "data, text, images, sounds, mask works, or computer programs" (NCCUSL, 1999, Section 102).

Even though the Act expressly states that opting into UCITA will not undermine the effects of any applicable consumer protections, it is difficult to gauge the impacts of this provision. As critics have noted, the opt-in provision is likely to have unforeseen consequences, in which products never contemplated by the drafters of the Act are brought within its scope. Lawyer Cem Kaner comments: "this provision threatens to swallow up the law of goods and undermine Revised Article 2.... In fact, as the opt-in provision is drafted, a loophole exists to allow any goods transaction to be put under UCITA by throwing a CD-ROM into the deal. For example, furniture could be sold with a CD-ROM giving cleaning or assembly instructions, allowing opt-in to UCITA" (Kaner, 1999a).

2. The Shift from Purchase to License, From Good to Service

UCITA confirms a fundamental ideological shift in the trade in information products. Pam Samuelson comments: "the paradigmatic transaction of [UCITA] is a license, that is, a limited transfer of rights to use information on stated terms and conditions. Contrast this with the dominant paradigm of the manufacturing

age, namely the sale of copies" (Samuelson, 1998, p.3). Of course, this is not an entirely new development, as software has been sold under license for some time. However, UCITA aims to iron out some of the legal uncertainties surrounding this method of transfer - especially mass market licenses -- and so solidify it.

The fundamental difference between purchasing a copy of software and receiving a license to use it is that a vendor has little ability to impose downstream controls on a purchaser's use of the product, while a licensor can impose many such controls on a licensee. As a licensee does not *own* a licensed product, a licensor can even prevent that licensee from using it at all in certain circumstances. A purchaser of software may enjoy full ownership rights subject to the requirements of applicable intellectual property laws. A licensee may only use the product subject to the terms of the license, which may be considerably more restrictive than intellectual property limitations.

The preamble to UCITA notes that: "A license is characterized by (1) the conditional nature of the rights or privileges conveyed to use the information, and (2) the focus on computer information, rather than on goods. A license differs from a sale or lease of goods in many ways, including in what the transferee receives by contract. One court stated: "[A] patent license agreement is ... nothing more than a promise by the licensor not to sue the licensee [even] if couched in terms of '[L]icensee is given the right to make, use, or sell X'". Images of a transaction that conveys ownership are not germane to licensing" (NCCUSL, 1999).

As UCITA promotes the use of mass market licensing, it signifies a shift in power between copyright and contract law. Several commentators have argued that this shift is unnecessary and detrimental to consumer interests (e.g., Samuelson, 1998).

3. "Shrink Wrap" and "Click Wrap"

Under UCITA, a mass-market license (a standard form contract used in dealings with the general public) will be enforceable against the licensee provided that the licensee manifested assent to it after having an opportunity to review it. The license will not be enforceable to the extent that its terms are unconscionable, contrary to fundamental public policy or in conflict with the parties' actual terms of agreement.

With shrink wrap and click wrap licenses, the licensee will not have an opportunity to review the license before purchasing the product. Under UCITA, this type of license is enforceable provided that the licensee knew the terms were coming, is given a right to return the product, cost-free, if s/he does not like the terms, and is reimbursed for any costs reasonably incurred in the process.

Critics charge that this extends current law by validating post-sale warranty disclaimers and remedy limitations, to the disadvantage of consumers (Kaner, 1999b). Some have suggested that costly litigation will be required to establish that terms are unconscionable or contrary to fundamental public policy, and that these concepts should be more clearly spelt out in UCITA (Kaner, 1998). Further, commentators argue that it undermines competition to validate terms not readily available to customers before purchase.

UCITA also allows for terms to be agreed by "reasonably configured" electronic agents. Pam Samuelson notes that due to the experimental status of this field, there is presently no way of judging whether an electronic agent is reasonably configured, and it is not even clear that agents will be used to contract (Samuelson, 1999). In this regard, the drafters appear to be too far in advance of current technological capabilities.

4. Consumer Protection

The drafters of UCITA argue that the Act preserves existing consumer protection statutes and does not alter federal consumer law: "a consumer is better off under UCITA than under existing Article 2 law for the sale of goods or current law for the sale of services" (Ring & Nimmer, 1999). Many commentators and consumer advocates disagree.

The additional protections that UCITA ostensibly offers consumers include a right of return and refund for consumers not satisfied with the license terms, and protection for electronic errors in contract formation.[9] Kaner responds that both of these protections are less favorable to consumers than current case law (Kaner, 1999a). Further, Kaner charges that by defining computer information transactions as sales of licenses rather than sales of goods, UCITA takes these transactions out of the scope of several consumer protection laws, including the federal Magnusson Moss Warranty Improvement Act and California's Song-Beverly Act (Kaner, 1998).

5. Warranties

UCITA includes warranties based on those applicable to the sale of goods under Article 2 of the UCC. It also includes implied warranties - for instance, if the licensee relies on the licensor's judgment as to compatibility of products, an implied license as to compatibility of products will be imputed. Any statement of fact or promise made by a licensor as part of the sale becomes an express warranty in the contract.

Several commentators have noted that UCITA does not require that software conform to documentation provided with it, as Section 402 (a)(1) provides that representations are only considered to be express warranties if they become "part of the basis of the bargain" (e.g., ARL, 1999). The result would be that users have no remedy if software fails to confirm to the manual provided with it.

The Software Engineering Institute protested to the NCCUSL Commissioners that UCITA "assumes that software products are inherently defective", and so creates a lowest common denominator quality standard in the industry (SEI, 1999). Another computer industry group, the ACM, adds that UCITA "makes it too easy for software publishers to avoid facing any legal consequences for defective software" (ACM, 1999).

6. Establishing breach

UCITA allows either party to refuse performance and cancel the contract in the case of a material breach by the other party. In situations where the purchase is not considered "mass market", the Act limits the definition of material breach by using a "substantial performance" rule - in essence, a party is unlikely to be considered to be in material breach of its contractual obligations if it has substantially performed them (s.701).

The situation would most likely arise if a software vendor delivers a product that does not conform to agreed specifications, or that differs from its documented capacity. In such a case, provided a court considers that the vendor substantially met the agreed terms, damages may be payable to the licensee but there will be no right to terminate. The drafters state that this is necessary to protect smaller software contractors against capricious corporate customers. Critics have argued that this undermines the "perfect tender" rule under Articles 2 and 2A of the UCC, which allows a buyer to refuse a product (and terminate the contract) if that product fails to conform to the agreed specifications (e.g., ARL, 1999).

The Association of Research Libraries regards the substantial performance criterion as a key defect of UCITA: "UCITA retains the perfect tender rule only in a "mass-market transaction" (which excludes most business software purchases by definition, even when off-the-shelf shrinkwrap software is involved). Purchasers who do not come within the restrictive "mass-market" definition are allowed to refuse defective software only if the defect can be considered a "material" breach of the contract requirements" (ARL, 1999).

7. Self help

Sections 815 and 816 of UCITA allow a licensor to exercise self-help remedies following cancellation of a license, including by repossessing information or electronically terminating access to that information, without judicial process. There are some limitations on this right; the self-help remedy is not available unless the licensee agreed to the part of the license that sets out this remedy, and the licensee must be notified at least 15 days before the remedy is exercised. Further, the licensor must not exercise the self-help remedy if such action will result in a breach of the peace, or involves a foreseeable risk of personal injury or significant physical damage to information or property other than the licensed information (section 815).

The self-help principles contained in UCITA have been borrowed from the Uniform Commercial Code's (UCC) treatment of leases and secured parties. For instance, section 9-503 allows an Article 9 secured party (a party with a secured interest in goods) to reclaim the relevant goods without judicial process following default, unless the parties have agreed otherwise or this would cause a breach of the peace. Similarly, section 2A-525 allows a lessor to repossess leased goods following the lessee's default.

The inclusion of such a provision in UCITA effectively gives a software producer an interest in licensed products equivalent to a secured or lease interest. This substantially extends the rights of such a party. UCITA proponents argue that the rights conferred under UCITA are substantially weaker than those under these provisions of the UCC: "Section 816 only applies [to] ... a material breach of contract or a breach that the agreement makes sufficient to cancel the license. [Under the UCC, an] Article 9 secured party, an Article 2 seller realizing on a retained security interest, and an Article 2A lessor, all are free to use their self-help rights upon any default, material or not" (Ring & Nimmer, 1999).

User groups are concerned about the potential of self help to imbalance contract negotiations and disputes. The Association of Research Libraries comments: "the Draconian remedy of electronic self-help provides the

licensor undue leverage in a dispute even if the remedy is not used. Faced with a crippling and possibly even fatal disruption of its business, a licensee could be intimidated into relinquishing license rights and setting up precedents for its further disadvantage" (ARL, 1999). This remedy appears particularly inappropriate where the dispute involves a level of complexity, and is not simply a matter of non-payment.

8. Criticism and Reverse Engineering

With the shift UCITA causes in the balance of power between contract and intellectual property, some consumer freedoms may not be appropriately protected. Kaner charges that under UCITA: "Users are not adequately protected against boilerplate terms purporting to limit their ability to publish results of tests of software or other criticism (found in several licenses) or to use software for "immoral purposes" (as a leading word processing product's license provides). When copies of software are sold, intellectual property law protects these user freedoms. It also makes unenforceable sweeping contract restrictions on quotation, communication of non-copyrightable information, or reverse engineering to fix problems with software or make products that work with other products" (Kaner, 1999a).

The possibility that UCITA will restrict reverse engineering activities has been particularly controversial. The President of the ACM charges that "UCITA allows publishers to ban reverse engineering by means of contractual use restrictions.... Software developers can freely reverse engineer mass-market products under current law. Without extensive litigation, over a span of many years, this right will be clouded by UCITA" (ACM, 1999). Richard Stallman asserts that such a limitation on reverse engineering would be "a disastrous obstacle for development of free software" (Stallman, 2000).

9. Duration, Transferability and Breadth of License

If the parties do not agree on the length of a software license, UCITA assumes that it will endure for a "reasonable time", subject to termination at will by either party (Section 308). In its list of criticisms of UCITA for business users of software, the Association of Research Libraries notes that "this is contrary to the currently prevailing commercial expectation that if the licensor wishes to limit the duration of the license, the time period must be stated in the license" (ARL, 1999).

UCITA also appears to change the existing legal position regarding the transfer of software licenses, which is a particular issue in the case of corporate mergers or software financing arrangements. Section 503 provides that a licensee may transfer its interest unless this would materially increase the burden on the licensor or materially impair its expectations. In practice, this is likely to require consultation with the licensor in the majority of cases, or else the licensee will run the risk of legal action. Pam Samuelson comments: "The UCITA transfer-of-license-rights rules are at odds with traditional principles of trade secrecy law. Since many, if not most, licenses have express 'no transfer' provisions, UCITA's default rule that the licensor's permission would be required when the license is silent runs counter to downstream parties expectation that a transfer is effective" (Samuelson, 1999).

Another restriction under UCITA relates to the number of people authorized by a license to use software. The Act indicates that if the license does not expressly limit the number of people who may use software under the license, it "permits a number of users which is reasonable in light of the informational rights involved and the commercial circumstances existing at the time of agreement" (s.307(c)). Critics have complained that this involves a change in practice that is detrimental to consumers: "it is widely understood that if the licensor intends to restrict the number of users, the license must state the restriction" (ARL, 1999).

10. Choice of Law and Forum

UCITA allows the parties to the contract to choose any legal system to govern their agreement - there need not be any natural nexus between the governing law and the transaction. Where there is no agreed choice, the law is either the law governing the vendor's principal place of business or, where a tangible product is delivered, the state of delivery. By contrast, under UCC Article 2, the agreement of the parties on this point is unenforceable unless they select a jurisdiction with a reasonable relationship to the transaction. UCITA also allows the parties to choose a judicial forum, provided that the choice is not "unreasonable and unjust".

Numerous commentators have asserted that these provisions disadvantage consumers, especially small consumers. In a mass-market context, licenses are almost never negotiated, so consumers are required to comply with the vendor's selection of law and forum. Consumers are very unlikely to scrutinize these provisions before entering into a licensing contract, and would only become aware of this issue when considering taking legal action. The cost and inconvenience of taking action in a foreign jurisdiction may well

discourage smaller consumers from pursuing their rights. To address this concern, Kaner argues that mass-market customers should be allowed to sue in their own state for smaller sums or where no forum selection is made in the contract (Kaner, 1998).

PART 3: GOVERNANCE IMPLICATIONS OF CODE LICENSING

There has been extensive discussion of open and closed code in the literature. In this paper we use UCITA as the representative of closed code, as it is the model created over a term of many years by representatives of proprietary software producers.

This section examines the landscape created by the open and closed code frameworks. We discuss the governance implications of the elements of open code. We present the view that governance consists of code plus a licensing framework, and then move to discuss the governance implications of the UCITA license. We conclude that these are models of vastly different economic systems, and we comment on the distinctions.

Open Code

First consider the definition of open code in the context of governance. What model of governance does open code present?

Free redistribution

For many people, free distribution (in the "free beer" sense) is the core feature of open code. But it is important to understand that free distribution does not preclude paid distribution. Programmers are entitled to add value to open code and then charge for that value. For example, there are fee-paying GNU-Linux distributions that include a GUI installer and thirty days of installation assistance. The alternative free distribution does not include the installer or any assistance. Further, RedHat offers a constant update line whereby patches are offered for any vulnerabilities, free, to those who have purchased the code. There is nothing in the free redistribution requirement to prevent versioning or value-added services.

In the governance sense, a parallel could be drawn between the free redistribution requirement and the minimal "welfare" accommodations offered by the state for many elements of life: food, shelter, and health care. In both cases there is arguably basic functionality that should be denied to none, although the quality and associated service may be considerably less than the market usually provides.

However, this is a flawed analogy, as it compares real and information goods. Because of code's status as an information good, a better argument is that free redistribution of code is comparable to the provision of basic democratic information. Thus this freedom can be seen as the right to access governance information, and is perhaps as basic as the right to vote in a traditional context. The information embedded in code contains the keys to powerful systems of control and authority in a virtual world. Open code advocates would argue that access to that information is essential to full and free citizenship in that environment (though of course, many "regular users" would not require the full degree of information that might seem essential to a programmer).

This approach suggests that freedom of code is the fundamental characteristic of open code, preventing the other characteristics of code from being denied by prohibitive pricing. Free distribution is enfranchisement, not welfare, in the open code world.

Source availability

Source availability is described by some (e.g. Lessig, 1999) as the most fundamental of all open code requirements. The availability of code offers the potential but not the guarantee of transparency in the governance sense.

In engineering, a system is transparent when it works so well that it is invisible to the user. By contrast, in governance a system is transparent if all steps can be examined and nothing can be hidden from the examiner. Thus open code can be seen as an analogy to the various sunshine laws, which require that government advisory bodies open meetings and records. Code availability is structural transparency.

Derivations

Derivations are a hotly contested issue in some open code environments. The debate echoes environmental arguments over the national heritage. Is the ideal goal of the national heritage to ensure it remains unchanged for all generations? Or is the heritage something to be modified according to maximization of profits? In the

United States, this conflict has existed over physical lands through out history, as exemplified by the Homesteading Act and various laws granting mining and logging rights on national lands.

Junior programmers declare that restrictions on the sale of derivations they have produced using open code severely hinder their ability to make a living (e.g., www.slashdot.org). Under the BSD License, derivations can be proprietary, as the new Mac OS has so amply illustrated. With Mozilla, only Netscape/AOL can create proprietary versions from the original.

So what of the information that is commonly owned? Is it a national asset to be used only with an expectation that all advances be returned, or is it raw material ripe for harvest? To the open source view, either case could be true, some code can be made proprietary and some remain free. To the free software view such privatization is theft. The core value seems to be that proprietary derivations are the privatization of the public commons.

Integrity

Integrity maintains the trustworthiness of the code base. Consumer protection regulation is embodied in integrity. According to the licenses with integrity requirements, any changes should be documented and redistributed with the original code. Integrity can be seen as a high level of control on altering the governance embodied in code, as any changes must be well documented and explained. In this way integrity provides the enforcement of claims of authenticity, along with transparency and openness.

Non-discrimination & non-specificity

Non-discrimination means that there is no group or application that may not use the code. Interestingly enough, this prevents code being limited to non-safety-critical systems. For example, the library in the first release of Java was not thread-safe. Thus the controls on safety-critical applications were arguable the socially responsible action. Yet it could be argued that the purpose of the non-discrimination clause is to prevent the release of such code, and that software should not be released in such a case.

However, this argument is flawed. It is belied by the articulation of "release early and often" by open source advocates as a whole. Rather we would argue that non-discrimination prevents open code from becoming involved in interline disputes, and prevents one element of code (e.g., a mailer) from becoming mired in some other conflict (say, between operating systems). With non-discrimination there can be ferociously competing camps at particular level or function in open code communities (e.g., BSD v linux, KDE vs Gnome) but those releasing software not immediately concerned with that layer or function are required to be agnostic. In effect, it prevents mandatory bundling.

Similarly, non-specificity prevents the rights attaching to code from being limited to any particular distribution. Again, it removes the temptation to discriminate and promotes an environment of equality of opportunity and technological agnosticism.

Non-contamination

While non-discrimination and non-specificity prevent mandatory bundling, non-contamination encourages promiscuous bundling. Non-contamination requires that it be possible to release the software in conjunction with other software. It enables consensus and community to form or coalesce around any subset of software without requiring that other elements be included.

Code as Law

Stallman and more recently Lessig have presented a view of code as law (e.g., Lessig, 1999). This approach regards code as the prime governance mechanism in an information society. While the analogy has some strengths, the flaw in comparing code to governance is that each person can choose his or her own code. In a Lotus Notes shop dominated by Windows with no support for Macintosh, we wrote this article on StarOffice on a RedHat Zoot box. A U.S. resident cannot similarly choose to bind herself by the laws of Belgium and not those of the United States.

Because of the specifics of the open code framework, each person can opt-out, and the decision of one person to opt-out does not measurably affect the choices of another. These basic facts must be considered in evaluating the social and governance implications of open code.

In our view, it is code together with the relevant licensing regime that defines the scope of governance. In this

analysis, an open code framework is best compared with a libertarian view of governance. Under open code, adoption of a particular package of goods does not prevent adoption of another, because the existence of a liberal set of governance rules requires continued transparency, a balance of power, and open processes. Because there can be nothing coerced from the user community through enforced bundling, or sold through guile or lack of integrity, the image of governance presented in the open code scenario is one of minimal governance and maximal individual autonomy.

Having argued that the open code proponents are in effect seeking a highly libertarian form of governance, in particular a government of code without the power to enforce or obligate the user to take action, we now use the principles of code plus licensing as governance to examine UCITA.

UCITA

UCITA is a paradigm shift both away from open code and away from code as property. With UCITA, users never have ownership rights over code, and in fact can lose ownership rights over their own data. Thus the governance paradigm best suited to describe UCITA is that of the corporate collective. The use of "corporate" is correct, because there is no public ownership. "Collective" applies because the core concept of purchasing is replaced with one of permission from the permanent owners. No user can own code, and nor can a user own alterations to the code, should they be permitted. All ownership remains with the corporation, and all additions or alterations can be claimed by the corporation. Rights of use are temporal and constrained. Users entering data into a proprietary format have no right to obtain that data in another form. In fact, a user breaking encryption to obtain data that he or she entered would be a felon, if the encryption was implemented for the purposes of protecting the corporation's property.

The BSA continues to be enthusiastic about UCITA, claiming that it will provide "a roadmap for consumers and businesses alike that want to engage in e-commerce by simplifying, standardizing and modernizing the law ... [and providing] ... a clear, consistent set of uniform rules while enhancing consumer choice and protection" (BSA, 2000). By contrast, the free software movement argues that UCITA's adoption by individual States will be disastrous for free software developers. Stallman presents the main issues as follows (Stallman, 2000):

1. Free software developers do not use shrink-wrap licenses, so they will not be able to avoid the default position of strict liability. Further, they do not have the legal sophistication and resources of the proprietary companies, so are less capable of protecting their interests under the proposed system.

2. UCITA will apply to free software that was released some time ago and is still being distributed, such that the presumption of liability will apply. There is little that developers can do to protect themselves in this case.

3. UCITA will allow proprietary developers to prohibit reverse engineering of their products, which will make it even more difficult for free software developers to create software which inter-operates with the major commercial programs.

Consumer rights bodies are also troubled by the proposed law. The Federal Trade Commission noted that UCITA allows software companies to place "restrictions on a consumer's right to sue for a product defect, to use the product, or even to publicly discuss or criticize the product". The FTC adds that "UCITA departs from an important principle of consumer protection that material terms must be disclosed prior to the consummation of the transaction". It goes on to say "we question whether it is appropriate to depart from these consumer protection and competition policy principles in a state commercial law statute" (FCC, 1999).

UCITA clearly addresses issues that go to the core of governance. In essence, the most basic conflict is the ability to examine the product or licensed service purchased. UCITA forecloses this possibility by being based almost entirely on object code. Secondly, UCITA addresses the core issue of speech by enabling restrictions on customer complaints. UCITA challenges concepts of citizenship by enforcing a choice of law and forum. In essence this would mean that a citizen of one nation, purchasing from a company incorporated in and resident in that nation, could be required to abide by the rights provided consumers in another nation.

Finally, UCITA alters the fundamental question of autonomy with controls on reverse engineering and self-help. There is a tradition of the home as inviolate, under which even the state is only allowed entry to the home under highly controlled circumstances. Self-help allows corporate access to personal electronic spaces, physically within the home, in order to enforce contract provisions. When viewed through the lens of governance, UCITA is radical indeed.

Open code proponents argue for a libertarian form of government with the option to opt-out. Free software

proponents argue for a more controlled form of libertarianism, where anyone can opt out but cannot limit another's ability to opt out. UCITA creates an environment without the ability to opt out from decisions. UCITA essentially moves us towards a world of code as law, since it removes the essential difference between code and law -- the ability of an individual to make his or her own rules. UCITA, in terms of governance paradigms, offers a quasi-totalitarian alternative where there is no opt-out, and reduced rights of criticism and privacy. We leave to the reader consideration of the implications of adopting the various regimes considered here, not only with respect to governance, but with respect also to the ability to choose to adopt code as law, or let code remain a source of ideas, and an object of personal choice.

FOOTNOTES

1. Serena Syme, B.S. LLB(Hons) University of Melbourne; Masters in Public Policy, Kennedy School of Government, Harvard University.

2. L.Jean Camp, PhD (Engineering and Public Policy) Carnegie Mellon, Associate Professor of Public Policy, Kennedy School of Government, Harvard University.

3. There is a subtle difference between the terms "open source software" and "free software". This paper will not use the two terms interchangeably, but will refer to "open code" when a point applies to both. The term "free software", as coined by Richard Stallman of the Free Software Foundation, draws attention to the liberty and rights attaching to the code. The term "open source software" was developed by Bruce Perens to assist in making the free software concept more attractive to commercial interests, and to eliminate confusion over the two meanings of "free" (it is often noted that free software is free as in speech, not free as in beer). The open source movement places less emphasis on the social imperative of freedom and more on the commercial benefits of openness. Further, the open source movement has certified the Apple APSL license as being source code, while the FSF rejects any suggestion that it is a free license (http://www.fsf.org/philosophy/free-software-for-freedom.html).

4. Some further comments are helpful on the issue of non-discrimination. The condition may seem unduly restrictive, but an example might help illustrate the benefits of this approach. Perens describes a license issued by the Regents of U.C. Berkeley preventing particular software developed there from being used by the South African police (Perens, 1999). That restriction was perhaps understandable during the period of apartheid, but no longer has the same relevance. Yet the license persists and continues to bind licensees from sharing their code with that category of user. It is true that the non-discrimination condition might prevent the use of open source code in the development of some restricted kinds of software, such as classified software, but the open code movement considers that a necessary evil.

5. BSA worldwide members include Adobe, Autodesk, Bentley Systems, Corel Corporation, Filemaker (Europe), Inprise(Asia), Lotus Development, Macromedia(Asia), Microsoft, Network Associates, Novell, Symantec and Visio. Additional members of BSA's Policy Council include Apple Computer, Compaq, IBM, Intel, Intuit and Sybase: http://www.bsa.org/about/index.html.

6. The NCCUSL consists of commissioners representing all 50 states, and its goal in preparing uniform laws is to provide consistency in laws from state to state. If a uniform law, such as UCITA, is approved by the NCCUSL, each state commissioner then introduces it as a bill in his or her own state's legislature. The NCCUSL passed UCITA in July 1999, despite opposition from a wide variety of interest groups.

UCITA critics include: The Federal Trade Commission, the Attorneys General of 26 states, the Institute of Electrical and Electronics Engineers (IEEE), the Association for Computing Machinery (ACM), the American Society for Quality, the Independent Computer Consultants Association, the Software Engineering Institute, the Free Software Foundation, the American Committee for Interoperable Systems, the Electronic Frontier Foundation, the Consumer Federation of America, the Consumers Union, the National Consumer League, the United States Public Interest Research Group, the American Library Association, the American Law Institute and the American Intellectual Property Association: see http://www.4cite.org/band1p.html . A full list of UCITA critics is maintained at http://www.jamesshuggins.com/h/tek1/ucita_opposition.htm.

In March 2000, Virginia became the first State to sign UCITA into law -- it will come into effect in that State on July 1, 2001 (http://www.nccusl.org/pressreleases/pr3-15-00.htm). The Act passed the Senate unanimously, and passed the House with only one dissent (http://leg1.state.va.us/cgi-bin/legp504.exe?001+sum+SB372CH).

A UCITA Bill was also passed by the Maryland legislature, and came into effect in October 2000: http://mlis.state.md.us/2000rs/fnotes/bil_0009/hb0019.rtf. Another UCITA bill is pending before the D.C. Council and its Committee on Consumer and Regulatory Affairs. UCITA will also be introduced in the Arizona, Illinois, Maine, New Jersey and Texas legislatures this year (NCCUSL, 2000).

7. Computer information is defined very broadly: "'Computer information' means information in electronic form that is obtained from or through the use of a computer, or that is in digital or equivalent form capable of being processed by a computer. The term includes a copy of information in that form and any documentation or packaging associated with the copy" (s.102).

8. This distinction can be complex. The Reporter's Notes accompanying UCITA indicate that "In most cases, if goods and computer information are in a transaction, good-based rules apply to the goods, but this Act applies to the computer information.....[Exception 1:] This Act applies to goods that are a copy, documentation, or packaging of the computer information [eg. the computer diskette or CD]....[Exception 2:] If the computer information is embedded in and inseparable from goods that are sold as goods...., [UCITA] applies to the computer information if the goods in which the information is embedded are a computer or a computer peripheral.... In other cases of embedded information, this Act does not apply to the information unless giving the purchaser the attributes of the computer information is a "material purpose" of the transaction.....When that occurs, other (goods-based) law governs the goods, but this Act governs the computer information". (UCITA, Reporter's Notes on Section 103: http://www.law.upenn.edu/bll/ulc/ucita/citam99.htm).

9. Under UCITA, a signature or its electronic equivalent is the means of authentication. A party relying upon the relevant authentication has the burden of establishing attribution of the authentication. Consumers who make errors in automated transactions (such as by mistakenly buying software) are not bound by those errors as long as they promptly notify the other party and return any "computer information" they received in the erroneous transaction.

REFERENCES

ACM, 1999, ACM letter on UCITA, www.acm.org/usacm/copyright/usacm-ucita.html, last viewed May 2000.

ARL, 1999, Association of Research Libraries, Summary of Key Problems with UCITA for Business Users of Software by Principal Financial - Fall 1999, http://www.arl.org/info/frn/copy/keyprobs.html, last viewed May 2000.

Behlendorf, 1999, "Open code as a Business Strategy" in C. DeBona, S. Ockman and M. Stone (eds) *Open codes: Voices from the Open code Revolution*, O'Reilly, 1999.

BSA, 2000, UCITA Signed Into Law in Maryland, http://www.bsa.org/usa/press/newsreleases/2000-04-25.132.phtml, last viewed April 2, 2001.

Free Software Foundation, 2000a, Various Licenses and Comments About Them, http://www.fsf.org/philosophy/license-list.html, last viewed December 21, 2000.

Free Software Foundation, 2000b, Non-Copylefted Free Software, http://www.fsf.org/philosophy/categories.html#Non-CopyleftedFreeSoftware, last viewed April 2, 2001.

Free Software Foundation, 2000c, The BSD License Problem, http://www.fsf.org/philosophy/bsd.html, last viewed April 2, 2001.

Free Software Foundation, 2000d, The Artistic License, http://www.fsf.org/philosophy/license-list.html#ArtisticLicense, last viewed April 2, 2001.

FTC, 1999, Bureau of Competition and Policy Planning office letter to Mr. John L. McClaugherty, Chair, Executive Committee, National Conference of Commissioners on Uniform State Laws, July 9, 1999: http://www.ftc.gov/be/v990010.htm, last viewed May 2000.

Hammerly et al, 1999: J. Hammerly, T. Paquin, S. Walton, "Freeing the Source" in C. DeBona, S. Ockman and M. Stone (eds) *Open codes: Voices from the Open code Revolution*, O'Reilly, 1999.

Kaner, 1998, Comments on Article 2B, October 1998, www.badsoftware.com/kanerncc.htm, last viewed May 2000.

Kaner, 1999a, Letter to NCCUSL, June 1999, http://commons.somewhere.com/rre/1999/RRE.UCITA.formerly.Artic.html, last viewed May 2000.

Kaner, 1999b, Objections to UCITA, July 1999: www.badsoftware.com/debate.htm, last viewed May 2000.

Lessig, 1999, *Code and Other Laws of Cyberspace*.

NCCUSL, 1999, Uniform Computer Information Transactions Act, http://www.law.upenn.edu/bll/ulc/ucita/citam99.htm, last viewed April 2001.

NCCUSL, 2000, UCITA fact sheet, http://www.nccusl.org/uniformact_factsheets/uniformacts-fs-ucita.htm, last viewed April 2001.

Perens, 1999, "The Open code Definition" in C. DeBona, S. Ockman and M. Stone (eds) *Open codes: Voices from the Open code Revolution*, O'Reilly, 1999.

Ring & Nimmer, 1999, Series of Papers on UCITA, http://www.ucitaonline.com/docs/q%26apmx.html#ES, last viewed May 2000. It should be noted that the authors of these papers were involved with the drafting of UCITA; Carlyle Ring was the Chairman of the NCCUSL Drafting Committee on UCITA, and Professor Nimmer was the Reporter to the Committee.

Samuelson, 1998, "Legally Speaking: Does Information Really Want to be Licensed?", *Communications of the ACM*, September 1998, http://sims.berkeley.edu/~pam/papers/acm_2B.html, last viewed May 2000.

Samuelson, 1999, Letter to NCCUSL, http://www.jamesshuggins.com/h/tek1/ucita_psam_19990709_letter.htm, last viewed May 2000.

SEI, 1999, Letter to NCCUSL from SEI, www.badsoftware.com/sei.htm, last viewed May 2000.

Stallman, 1999, "The GNU Operating System and the Free Software Movement " in C. DeBona, S. Ockman and M. Stone (eds) *Open codes: Voices from the Open code Revolution*, O'Reilly, 1999.

Stallman, 2000, 'Why We Must Fight UCITA', *Linux Today*, Feb 6, 2000: http://linuxtoday.com/stories/15948.html, last viewed May 2000.