Journal of Information, Law and Technology

Code as Embedded Speech, Machine, and Service

L. Jean Camp
Assistant Professor, Kennedy School of Government
Harvard University, USA
jean_camp@harvard.edu
<*http://www.ljean.net*>

Serena Syme
Masters of Public Policy
Kennedy School of Government
Harvard University
scsyme@hotmail.com

This is a **refereed** article published on: 2 July 2001

Citation: Camp L J and Syme S, 'Code as Embedded Speech, Machine, and Service', 2001 (2) *The Journal of Information, Law and Technology (JILT)*.
<http://elj.warwick.ac.uk/jilt/01-2/camp.html>

Abstract

What is code and how should it be governed? This article frames the question by considering the various forms of governance currently applied to code: copyright, patents, embedded speech, and trade secrets. The current governance models imply that code is a service, a document, and a machine. True, physical materials can be arranged to be all of these things, yet there is little legal difficulty in distinguishing which model is appropriately used for which item. Thus the mental models of code and of different types of underlying the legal definitions of code are confused. Thus, we being our work by describing both code and the protection modes which can be used for code (at a high level, of course). Then, in the context of both recent court decisions and private contractual models for code, we work to fit code in the context of the governance frameworks. We conclude with an overarching theory of code which argues for the continuation of all the models of protection, yet offers a manner of distinguishing which models should be applied to which instantiations of code. In short, we conclude that source code should best be governed as a form of speech and object code should be treated as a class of machine. We conclude that customer code sold to one customer treated as a professional service, possibly with requirements for source availability.

Keywords: Software Protection, Code, Governance of Code, Theory of Code, Models of Protection for Code, Source Code, Object Code, Intellectual Property, Copyright, Patents, Embedded Speech, Machine, Service, Trade Secrets, UCITA.

1. Introduction

What is code? When is the procurement of software the licensing of a service, the purchase of a product or the free exchange of ideas? Currently software - including source and object code- fits under all three rubrics. Industry practices, intellectual property regimes, the aims of the market, and the traditions of science together form an incoherent and almost certainly unsustainable regulatory patchwork for computer code. In this paper our goal is to answer the question, 'What is the appropriate regulatory regime for code?'. In order to do so rather we seek to frame the question more fully than in past literature for the specific case of code.

Previous work has focused on the ethical implications of code1, a specific self-regulatory approach2, industry practices3, or information goods as a whole4. 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 intellectual property protection specifically for code. We consider the use of copyright by open code, the lack of control associated with code in the public domain, licensing as proposed by UCITA, and the current use of patents as code control mechanisms. Part 1 provides a brief outline of the various forms of intellectual property protection that are applied to software: copyright, patent, trade secret and contract protection. In Part 2 we compare and contrast the protection offered by each of these schemes. Those familiar with intellectual property law can reasonable skip this section. Part 3 is a discussion of the types of code. Those familiar with computer science can reasonably skip Part 3. In Part 4 we build upon

this foundation by addressing how intellectual property protections interact with open code licenses, proprietary licenses, and licenses under UCITA. After having classified code according to its mode of governance, we close by arguing that the mechanisms used to govern speech (the First Amendment, traditional copyright and trade secrets protection) are the correct mechanisms for the governance of code. In our final conclusion in part we offer a preliminary framework which would treat source code as embodied speech, mass market object code as product, and custom software as professional service.

2. Intellectual Property Protection

The previous parts, described licenses and their constraints, and the impact of UCITA. This section summarizes the previous discussion for simplified comparison and reference. It then briefly outlines the types of intellectual property protection used in each license.

2.1 Types of Intellectual Property

Different types of software, that is software protected under different property rubrics, depend on different elements of the intellectual property legal structure. There are four rubrics for intellectual property: copyright, patent, trade secrets, and contract. There is no argument over the need for some type of intellectual property protection for code.

2.1.1 Copyright Protection

Copyright is a form of protection granted under Title 17 of the U.S. Code to those who create 'original works of authorship' that are fixed in a tangible form of expression. Under Section 106 of the Copyright Act 1976, the owner of copyright usually has the exclusive right to reproduce the work, make derivative works based on it, distribute, perform or display the work. The owner also has the exclusive right to authorize others to exercise these rights. Copyright protects the physical embodiment of an artistic or creative work. That is, it will prevent others from copying the actual form of expression selected by the author, but will not protect the ideas, thoughts or processes underlying that expression. Section 102 of US Code Title 17 indicates that copyright subsists in 'original works of authorship fixed in any tangible medium of expression', including literary works, musical works, dramatic works, pantomimes and choreographic works, pictorial, graphic, and sculptural works, motion pictures and other audiovisual works, sound recordings and architectural works. An important exception to the property rights that copyright confers on a creator is the doctrine of 'fair use'. Section 107 provides that copyright is not infringed by the fair use of a work protected by copyright 'for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research'. Copyright protection will endure for 70 years after the death of the author. Copyright subsists automatically in the work - there is no need for the author to register the work or take any other steps to obtain the protection. However, an author can abandon the copyright protection, if he or she so chooses. A work for which copyright protection has been abandoned or has expired is considered to be in the public domain.

Copyright and patent protection evolved in Western countries at the time of the printing

press, as an attempt to assure the free transmission of ideas while safeguarding the rights of the creator. In fact, the modern concept of authorship itself arose with the printing press5. These forms of a protection are a strictly limited monopoly, a form of bargain with inventors, artists, and authors; promising remuneration via a legal right to exclude.

Copyright provisions are careful not to encroach on constitutional protection of free speech. Litman notes that copyright permits free communication of facts and ideas while protecting an author's expression, so striking a balance between the First Amendment and the Copyright Act. Unlike patent protection, copyright protects the form of expression but not the concept behind the expression -- it 'leaves others free to communicate the ideas embodied in protected works, so long as they do not appropriate the form in which those ideas were expressed6.

The Internet's has created challenges to copyright, particularly in terms of copying and material embodiment7. Copyright's dependence on material embodiment and the incentives it offered worked well when the works protected by it were physical. But the Internet poses a number of challenges to the copyright system:;The technical reality of the Internet is that the works displayed on it are frequently copied, often without deliberate human effort. In a digital system to view or transfer a work requires making copies. Each time a browser caches a web site, that web site and the information on it is copied. Buffering requires information or images to be automatically loaded to one location and then transferred to another one - which again is copying. Johnson-Laird describes the Internet as a global copying machine:

> 'everything on the net is a copy of something -- else if it is just a local copy of a World Wide Web page on a local Web server'8.

How should a copyright system deal with these 'involuntary' copies?

Technological changes have resulted in economic changes, at the least changes in business models. The economy of the Internet suggests that there is value in the rapid dissemination of information. Dyson claims that the Internet changes the fundamental economics attached to the selling of information:

> 'because it allows us to copy content essentially for free, the Net poses interesting challenges for owners, creators, sellers, and users of intellectual property. In this new world of the Net, it is easy to copy information but hard to find it'9.

Dyson argues that in this new, copy-centric environment, content will tend towards being free, because it is the free content that will be rapidly disseminated and noticed. She adds that under these conditions:

> 'the likely best course for content providers is to exploit that situation, to distribute intellectual property free in order to sell services and relationships'.

This as yet not tested or proven 'new economics' suggests that fundamental changes in

intellectual property restrictions may be needed.

As the works copyright seeks to protect lose their physicality, copyright may begin to threaten freedom of speech. Copyright was designed to protect the physical embodiment of ideas. Now ideas can be rapidly disseminated without any physical embodiment, which makes them difficult to distinguish from speech. It follows that efforts to extend the scope of those laws so as to protect such insubstantial creations is likely to result in restrictions on freedom of speech.

This had lead to widespread consideration of the future of copyright on the Net. At one extreme, Barlow argues that copyright's ship is sinking beneath the waves of this new environment:

> 'This vessel .. was developed to convey forms and methods of expression entirely different from the vaporous cargo it is now being asked to carry. It is leaking as much from within as from without. Legal efforts to keep the old boat floating are taking three forms: a frenzy of deck chair rearrangement, stern warnings to the passengers that if she goes down, they will face harsh criminal penalties, and serene, glassy-eyed denial'10.

On the contrary, free software proponents argue that copyright is essential to protect the freedom of their work11. Unless code is appropriately copyrighted, users can 'capture' that code back into the private domain and apply stringent license conditions to it. Further, software companies argue strongly that copyright should continue to apply in on-line space, and be enforced more stringently. The Software and Information Industry Association (SIIA) notes that the fact that software creates unique problems for copyright:

> 'does not make it legal to violate the rights of the copyright owner'12.

Copyright applies internationally, and to code internationally. (In contrast software patents are a divisive international issue). Article 7 of the Berne Convention for the Protection of Literary and Artistic Works 1971 states that the term of copyright protection is 'the life of the author and fifty years after his death'. The World Intellectual Property Organization's 1996 Copyright Treaty clarifies the application of the Berne Convention to computer programs, among other things. Article 4 provides that:

> 'Computer programs are protected as literary works within the meaning of Article 2 of the Berne Convention. Such protection applies to computer programs, whatever may be the mode or form of their expression'.

The US Congress has recognized that there the current intellectual property regime does not naturally apply to code. The US Congress passed the Digital Millennium Copyright Act (P.L.105-304) in October 1998 to implement the 1996 WIPO Copyright Treaty. Among other things, the DMCA prohibits the circumvention of any effective 'technological protection measure' (such as encryption) used by a copyright owner to restrict access to its material13. Until recently, the Berne Convention's copyright duration of 'life plus 50 years' applied in the United States. However, the Copyright Term

Extension Act (the 'CTEA') passed in October 1998, extended the term of copyright protection by 20 years to 'life plus 70 years' for individual authors, and to 95 years from 75 years for corporate 'creators'. One motivation for the CTEA was the desire of copyright owners to obtain protection in the US equal to that provided by European countries. As long as US laws offered shorter protection than the European laws, US authors could not benefit from the full length of protection available in Europe13. But it seems that many of those arguing in favor of CTEA were also worried about losing copyright protection in lucrative properties. Disney, which was set to lose copyright protection over the Mickey Mouse character in 2004, made financial contributions to 8 of the Senate bill's 12 sponsors, and to 10 of the original House bill's 13 sponsors14. The CTEA attracted much opposition. For example, the Society of American Archivists expressed their disapproval to Congress, stating that the law 'disrupts the balance between public and private interests and will have a severe negative impact on the public's use of unpublished materials for teaching, scholarship, and research'15. The Association of Research Librarians describe it as an 'unfortunate law', and a group of small publishers and archivists of public domain material brought a constitutional challenge against CTEA.

The Eldred *v Reno*16 case was heard in the District Court, where Judge June Green granted summary judgment to the government on October 28, 1999. It is now on appeal. In our references to copyright in the remainder of this article, we are referring to traditional copyright protection, not copyright as amended by the DMCA and CTEA. The implications of the DMCA and CTEA do, in the opinion of the authors, tend to exacerbate the problems with the current incoherent governance of code. Yet the impact of the DMCA is yet uncertain, and is beyond the scope of this work.

2.1.2 Patent Protection

Just as the subject matter of copyright is a tangible expression of authorship, the subject matter of patent law is an invention. Title 35 of the US Code deals with patent protection - section 101 of that title defines an invention as subject to patent if the invention is any 'new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof'. The inventor or discoverer of such an invention may patent it, provided that:

- the invention is novel, which usually means that it was not disclosed in the 'prior art';
- publications or other documents relating to that field of endeavor;
- was not patented elsewhere more than a year previously (section 102);
- the invention has not been 'abandoned' by the inventor (section 102); and
- the invention is 'non-obvious', in the sense that it would not have been considered to be an obvious development by a 'person having ordinary skill

in the art to which said subject matter pertains' (section 103).

A person seeking patent protection must engage in a lengthy and complex application process that usually requires the involvement of professional patent attorneys to establish the validity of the patent grant. The United States Patent and Trademark Office hears patent applications and is the sole grantor of US patents. Patent protection endures for 20 years (section 154). However an individual can approach the US Patent and Trademark Office.

Patent protection is broader than copyright protection, in that it prohibits more than simple copying of the invention. A patent will be infringed if another person independently creates an invention that comes within the terms of the patent. So if Person A was to patent a left handed corkscrew with a 90 degree angle between the screw and the pull mechanism, for instance, any other person who produced a corkscrew matching this description would infringe the patent. This would be the case whether or not the infringing person had ever heard of or seen the patent or the patented device. Due to the breadth of this protection, innovation is promoted by ensuring that patented inventions are very narrowly described in the application.

One of the more controversial areas of patent protection is the patenting of computer algorithms. Until recently, the USPTO would not patent a business method. But following the 1998 decision in State Street Bank & Trust Co. v. Signature Financial Group which opened the way for these patents, software manufacturers and web companies have been extremely active in seeking protection for their 'inventions'. As an example, Amazon.com has obtained patents to protect it's '1-Click' shopping method17, and more recently, its affiliate program18. These companies argue that this protection is essential to safeguard their investment. Opponents argue against such patents on a range of issues:

- patent protection is not necessary to promote software innovation, and may well impede innovation19;
- innovation costs in this industry are substantially less than those in traditional patent fields such as pharmaceuticals20;
- patent searches (which are necessary to avoid infringement) are expensive, unreliable and prohibitively difficult for smaller companies21;
- independent reinvention, which is quite rare in some other fields where patents are active, is extremely common in the software industry22;
- business process patents require so little actual inventive work that they amount to a patent on an idea23;
- a 20 year term of patent protection is unreasonable in this fast moving environment24; and,

- the prevalence of patents restricts programmers' creativity and expression.

Others argue that patent protection software is sufficiently excessive as to create barriers to entry as well as innovation25. If there is no change in the law, it is reasonable to expect the number of web and software patents granted to increase exponentially.

2.1.3 Trade Secret Protection

Patent protection requires disclosure of the invention; the bargain that the State makes with inventors is that 20 years of protection is offered in return for this disclosure. Trade secret protection does not require disclosure, and for this reason may be relied on for especially confidential commercial formulae. The recipe for Coca Cola is the canonical example. Section 757 of the Restatement of Torts (1939) indicates that:

> 'a trade secret may consist of any formula, pattern, device or compilation of information which is used in one's business, and which gives him an opportunity to obtain an advantage over competitors who do not know or use it. It may be a formula for a chemical compound, a process of manufacturing, treating or preserving material, a pattern for a machine or other device, or a list of customers'.

To be eligible for this protection, the information or invention must be secret, and it must confer commercial advantage. Secrecy implies that it is not within the general knowledge of those working in the field, and also that the owner has taken steps to guard it. It generally implies that the information is not widely distributed. Unlike copyright and patent protection, trade secret protection generally occurs under State law rather than Federal law, and the protection offered is very limited. A trade secret holder is only protected from unauthorized disclosure and use of the trade secret, or from a third party obtaining the trade secret by improper means. It is not an infringement of this protection for a third party to reverse engineer the secret (i.e. determine the recipe for Coca Cola by working backwards from the taste of the drink), or to arrive at it independently (i.e. develop Coca Cola without having tasted it). Trade secret protection essentially guards against corporate espionage only. As for copyright protection, an owner does not need to take any special steps to rely on this form of protection (except by safeguarding the secrecy of the information). But unlike copyright and patent protection, trade secret protection can easily be lost - one or more unprotected disclosures of the information will usually be enough for a court to conclude that the information is no longer secret. The secrecy requirement means that this form of protection would not be relied on to protect open code software, as the provision of the code to users would essentially undermine the secrecy of that code. But trade secret protection could conceivably be used to prevent someone from entering Microsoft headquarters and stealing the Windows source code, or to punish a disgruntled employee who released that code to the world.

3. Code

Code is no more homogenous than any other information type, and in fact it is far more heterogeneous than any other information type. Computer code takes the same nomenclature as legal code and there are two other similarities. First, both legal and computer code attempt to structure inherently entropy life into formal conditional logical structures. Second, by virtue of this translation or encoding both are difficult to decipher for those without trained and both can appear encoded in the cryptographic sense to the untrained eye.

3.1 Forms of Code[26]

Computer code exists along a continuum. At one end is source code. Source code is optimized for human readability. Source code is high level code. In fact, the readability of this code has been decried by some[27] because using the new coding schemes does not require any fundamental understanding of computing or communications.

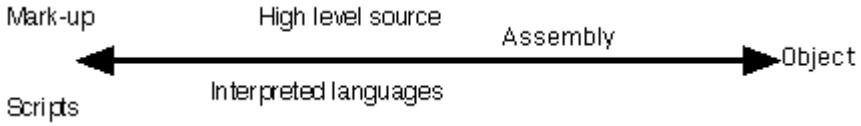

Figure 1: How code can be read

As shown in the figure above code can be of a form that is inherently human readable. Example of this are mark-up languages, such as the hypertext markup language used to format this document. There are technical means to prohibit the trivially easy reading and viewing of a document source, and methods for writing increasingly obtuse source code are proliferating. For example, popular Web-authoring documents use unnecessary Java calls or covert Web pages to Shockwave formats which cannot be easily read. However, markup languages are designed to be readable.

The same is true of scripting languages such as JavaScript and CGI[28] scripts. Such scripts are read (thus the name) each time the script is called. Scripting languages are stored and transmitted in source form.

Assembly is the original coding language. In assembly language humans had to use the tiny steps which a computer can understand, like moving number between registers, to write programs. For example, a subtraction instruction in assembly takes many lines of code. The computer must be instructed to read the input a, one by one. Each input that is read must be assigned an address in memory. Then the inputs must be copied to input locations of the Arithmetic Logic Unit which handles subtraction. Then the ALU must be told to subtract the numbers. Then the result must be stored in a particular location. Computers can understand only the most simple, short commands.

Grace Hopper (invented the concept of a forerunner to all modern compilers, and thus enabled the creation of high level languages[29]. Before her breakthrough work all code was written in binary, and in fact often implemented by cabling ports together. Today

even modern ‘assembly’ must be run through an assembler to be read by the machine. While the human programmer must make the steps easy for the machine, and write the steps in the correct order, the assembler must alter the steps into exact binary instructions which include, for example, exact addresses rather than the command to move a number to memory (or store) or to the ALU input.

The earliest code was all binary; of course, and thus clearly the most basic binary codes can be read. In these early codes the commands were implemented by women who physically linked nodes to create the binary ‘1’ of the commands. However, the codes produced today are orders of magnitude larger and thus more complex to read in binary form. For example, during World War II the effort of all the Allied mathematicians at Bletchley Park to create a coding to break German’s encrypting Enigma machine. The mathematical genius who determined how to break the machine using the binary computing power available at the time, Alan Turing, is now being honored by s 20,000 pound statue in Manchester’s Sackville Park. In contrast, creating high-level code to break the Enigma machine is a not uncommon undergraduate assignment at Carnegie Mellon, as the coding is far simpler and the available processing power far greater. The coding involved in creating a modern game, WYSIWYG word processor, or operating system requires higher level code and produces millions of functionally unreadable binary code. Thus the code written today is commonly subject to reverse engineering which observes the actions and interactions of the code with other elements of the machine; as opposed to attempting to read the code itself.

High level languages can be used for far more complex tasks than scripting languages, and are more efficient for multiple use on a single machine. High level languages can be compiled or interpreted. Compiled code is read directly by a machine, and is suitable only for a particular machine running a particular operating system. A compiler changes words to bits, of course, but more importantly it changes instructions which can be understood by a human(e.g., x+1)to instructions which can be understood by a computer. Interpreted languages similarly alter high level instructions to instructions which can be understood by a virtual machine. A virtual machine breaks the actions into the same set of long simple steps which a computer can understand. However, it uses interim address and names rather than actual hardware addresses (e.g. Arithmetic Logic Buffer 0 as opposed to a specific 16 or 32 bit address). Thus interpreted code can be run on any machine capable of supporting a version of the virtual machine. Java is interpreted code and the C family of languages is compiled.

The output of a assembler, compiler or interpreter is source code. Object code is machine-specific even if the machine is a virtual machine. The object code for a complex program, which may be written on a scale of 10^6 lines, cannot be read by humans. Object code can be read only be de-compiling, which is a painstaking process requiring as much artistry as engineering. De-compiling object code will not result in the same code as was entered into the compiler, but rather code which does the same thing as the original code. High level languages are optimized to be read by humans, in contrast to low-level or binary code is optimized for the machine.

3.2 Code as Speech

In addition to the arguments within the computer science community as to the appropriate nature of code there have been three significant court cases which hinge on the question of code as speech. One critical element of all three cases that the code in question in all these cases is source code. These three cases are *Junger v. Daley*[30], *Bernstein v. US Department of State*[31] and *Karn v. United States Department of State* [32].

The case of;*Junger v Daley* deals with the question of whether computer code is entitled to protection as free speech. The plaintiff, Cleveland Law School Professor Peter Daley, sought to publish examples of encryption code on his class web site. However, Export Administration Regulations prohibited the export of certain types of encryption, and this form of publication was deemed to be a prohibited export. The District Court rejected Professor Daley's arguments that free speech protection should apply to source code. Judge James Gwin concluded that source code should be regarded as a functional device, outside the scope of First Amendment protection. The plaintiff appealed, and the matter was heard by the United States Court of Appeals. That court's recent determination reversed the District Court decision, determining that:

> '[b]ecause computer source code is an expressive means for the exchange of information and ideas about computer programming... it is protected by the First Amendment'[33].

The written judgment indicates that the court was influenced by the fact that source code must be converted into object code before it can be executed by a computer, so the code cannot be regarded as mere functional executable code. It has 'both an expressive feature and a functional feature'. Further, the First Amendment does not require the speech to be comprehensible to the average person: 'a musical score cannot be read by the majority of the public but can be used as a means of communication among musicians. Likewise, computer source code, though unintelligible to many, is the preferred method of communication among computer programmers'[34].

The facts *Bernstein v. US Department of State* are similar to those of *Junger v. Daley*. Bernstein is a Professor of Mathematics, and Computer Science at the University of Illinois. As is now common in many universities, Professor Bernstein published the material for his courses on his web site. This material included cryptographic source code.

In *Karn v. United States Dep't of State* the plaintiff attempted to export a text on encryption. The book, *Applied Cryptography*[35] includes the written text of source code, which is now and has been classified as speech. The book also includes a disc which has the source code described in the book. The disc does not include a compiler. The book can be exported under current law, yet the disc which accompanies the book cannot. Karn sought to export the disc as well as the printed text.

The key element in all of these cases is the use of source code as expression, to communicate between people. The full implications of the application of the First

Amendment to computer code are yet to be explored. A regular right to free speech is not a right to plagiarize, so this change in the law is unlikely to impact upon programmers' rights with respect to the copyrighted computer code of others. The arguments with respect to the nature of code which encrypts has been the area where discussions of the nature or code have been most detailed[36]. Yet this debate has not extended to the discussion of the interaction of intellectual property and code, no doubt in part because the core of the argument has been the existence of First Amendment rights for those who communicate with source code.

With this basic understanding of code we can proceed.

4. Licenses and Legal Protection

Licenses are intellectual property mechanisms which address the distinct types of code and the protection for that code. Notice that within the computer science community the construct which we offer here has been implicit if unexamined in the private governance structure (i.e. licenses) for decades. Here we discuss the types of licenses at some length, to bring to light the options for governance of code embodies in each license. We also discuss the interaction between code types and intellectual property protection in each license.

Contract law regulates the relationship between the parties to the contract - it does not provide those parties with rights against the rest of the world. In that sense, contract is essentially a private matter, and courts will allow parties significant discretion to negotiate and agree on terms suitable to their situation. Courts may be inclined to override contractual terms where they are considered to be unconscionable, or if they contravene public policy. But where the parties are negotiating at 'arms' length' in a commercial context, the bargain and terms they reach are likely to be upheld by the law. The 'shrink wrap' and 'click through' licenses included with software are governed by the law of contract. By their nature, licenses convey rights to use software, but no proprietary rights in that software. As a result, the 'purchaser' of software under a license receives a variety of contractual rights (to use, copy or modify the software, for instance), but cannot be considered to own the software. The license can be revoked (such as if there is a fundamental contractual breach by the licensee), and this would generally terminate the licensee's rights in respect of the software.

This section considers how intellectual property protections interact with open code licenses, proprietary licenses and licenses under UCITA.

4.1 Open Code Licenses

Copyright protection is essential to the effectiveness of open code licenses. Once copyright protection is waived in relation to code, that code becomes public domain. As noted in the discussion of public domain software in Part 1, this means that the code can be removed from the public domain and used in proprietary software. Copyright is necessary to keep open code free or open. Notice that the copyright proposal proposed by

Zittrain37 and the less considered proposals by Barlow, would effectively kill open code licenses.

Why is copyright needed when the parties could simply rely on a license agreement? The existence of copyright creates a default position restricting all users' ability to copy or amend the software. If users want more extensive rights in relation to open code software, they must rely on open software licenses, which tend to be more permissive than copyright. Once they rely on these licenses for the benefits they confer, they must also comply with the license restrictions and any disclaimers or warranty exclusions in the license. This approach offers benefits to both parties. If copyright were not applied, it would be much more difficult to establish that a person using freely distributed code had entered into a license agreement and was bound by its terms and conditions. Many members of the open code software community reject software patents, charging that they restrict expression and creativity and are generally bad for business38. Open code licenses do not rely on patent protection, although they may be able to interact with patent protection. As open code distributions necessarily include a distribution of the underlying source code, trade secret protection will not apply to that code.

4.2 Proprietary Licenses

A proprietary software license usually reserves all applicable intellectual property protection attaching to the product. For instance, Microsoft's End-User License Agreement for Windows 98 (the 'MS License') states:

> '[t]he SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold'.

Article 4 of the MS License goes on to state:

> All title and intellectual property rights in and to the content which may be accessed through use of the SOFTWARE PRODUCT is the property of the respective content owner and may be protected by applicable copyright or other intellectual property laws and treaties. This [license] grants you no rights to use such content. All rights not expressly granted under this [license] are reserved by MS and its suppliers (including Microsoft Corporation).

The software producer is also likely to attempt to restrict consumer rights conferred by intellectual property laws to the maximum extent possible. As an example, Article 2(e) of the MS License provides:

> 'You may not reverse engineer, de-compile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is;expressly permitted by applicable law notwithstanding this limitation' [italics added].

The effect is to limit any rights that might be implied under the law. However, at present, there is some doubt about the enforceability of these licenses, and so software producers

may be left to rely on their intellectual property rights alone. As source code is usually not distributed with proprietary software, the software producer may be able to rely on trade secret protection in relation to any unauthorized disclosures of this code.

4.3 UCITA

A license under UCITA is simply a commercial contract, which depends on the parties' abilities to enter into such a contract. UCITA cannot be used to extend a licenser's intellectual property rights, and expressly recognizes preemption by copyright, patent, or other federal intellectual property law. Under UCITA, proprietary software licenses are likely to continue to reserve all intellectual property rights and restrict consumer rights under those laws to the maximum extent possible. As these licenses are more likely to be enforceable than existing mass market software licenses, this approach can be expected to offer software producers at least as much protection as existing proprietary licenses and possibly more protection.

5. Conclusions: Code as Speech, Product or Service

Proprietary code is currently sold as a product. Code is purchased as the artefact on which it is embodied, usually a CD ROM. Code is packaged and sold as a product to the home consumers. Similarly, the software patenting scheme views code as an invention, as a functional device that operates to produce an outcome. UCITA would change this practice by having code be controlled as a service with extreme variation in the licensing terms. The closest match to UCITA today is in professional services - lawyers or consultants who can be held liable only for negligence or misrepresentation. UCITA offers the view of code as an active service performed by the company on the consumer's computer. Open source and free software offer a view of code as speech, which must be protected to allow it to be freely shared, examined and dissected. By contrast, software in the public domain has no protection, and thus cannot be said to be governed. This analysis gives rise to four implicit models of code governance; code as a product or functional invention, code as professional service, code as speech and ungoverned code. The table below and our argument above supports the application of these three models by showing the correspondence between the intellectual property mechanisms used for code governance and the mental model that we propose here.

Implicit model of Code	**Open code/ free licenses**		**UCITA**	
Copyright	**Patents**	**Trade Secrets**	**Public Domain**	
Product or functional invention				X X

Professional service		X			
Embodied Speech	X		X		
Ungoverned Code					X

Table 1: Correspondence between models of software and forms of legal protection.

If the court is correct in Junger *v Daley,* arguably the only sustainable model of code

governance is that which recognizes code as embodied speech. Our framing suggests that the ruling in;*Junger v Daley* may have far-reaching (at least) philosophical and (at most and unlikely) regulatory effects. At its extreme this ruling may be seen to imply that First Amendment protection, copyright and trade secrets - the only mechanisms useful for that which is embodied speech - are the only feasible mechanisms for the governance of code. When code is the specialized speech that governs our world, as Lessig would argue, this degradation of dialogue is costly

It is critical that the foundations for the protection of code be thoroughly considered with a broad view encompassing not only intellectual property but also all current experiments in code governance. By describing the essence of code as well as the mechanisms of its governance we have concluded that source code should be governed as embodied speech, object code governed as product or functional invention. The third option - professional service - should be restricted to code developed for a single customer as specified by the customer under an individually negotiated contract. An example of such a situation may be the efforts by software engineers to prevent any difficulties resulting from the two-year field date, widely known as the Y2K problem. Other examples may be the conversion of databases or custom interfaces or web designs.

Legal Protection	**Conceptual Model**	**Protecting**	**Application to Code**
Patent	Code as product or functional invention	specific implementations of idea, practical innovation	Object code
Copyright	Code as speech	expression of idea	Source code
Professional service	Code as customized service	results of professional judgment	consultants, Custom produced code
None	Ungoverned code	information which wants to be free	Public domain

Table 2: An Intellectually Coherent Application of Law to Digital Information Property

Such a reorganization of intellectual property law would maintain the governance of code, reward inventors, allow innovation and prevent restrictions of speech. The proposal for reconsideration of intellectual property has the advantage that it fits the mental models as embodied in different licenses and proposals with respect to the governance of code.

While this is an innovative proposal it is neither as radical as any who would propose a single model for all code, from information wanting to be free to information which is licensed, nor would it prohibit the models for software market as they exist today. It is in concert with the fundamental concepts of intellectual property and business practice. Both the free software and open source approaches to code governance fit within our proposed implementation. In the case of open code or free software those who choose to install software have changed the form from embodied speech to machine. Just as a producer of a textbook will not be held liable if a student were to harm him/herself when building from its directions this would not increase the liability of those providing source code.

The patent system is not fundamentally at odds with this framing. It would require that patents be limited to a particular instantiation of source code, rather than a concept which

can be coded. Yet innovations as implementations in software would remain protected by patent. The more interesting issue would be the question of functional equivalence in software. This may produce exactly the same negative result as overly broad software patents is said to have produced: a unknowable web of constraints on the individual innovator. Alternatively this may allow any significant innovation to occur while providing patent protection for the most fundamental innovations. Depending on the construction of the concept of equivalence for functioning software, the patent protection on software as a machine could be broad enough to have prevented the creation of Microsoft Excel from Lotus 1•2•3, Microsoft's Explorer from Navigator. Despite the findings of fact that Explorer was initially technically inferior to Navigator and adoption was forced by monopoly power, the extension of the patent system to cover any GUI browser would be a considerable and potentially dangerous extension, rather than contraction, of patent protection. Under the current law, the final competitive choice for Netscape was to release its code. Under the proposed framing, such a release would entail trading the more stringent protect ion of copyright for the lower liability given instructions in the printed word. Maintaining trade secrets would not be altered by this reorganization of intellectual property law.

There are two perspectives which would be not be possible under this proposed regime. The first is the declaration that information is inherently free, and that any protection is ill-considered. Both as authors and individuals who would purchase high quality information, we do not believe that all information wants to be free[39]. The second is the model of code as implemented in UCITA. UCITA is in direct opposition to this proposal as UCITA treats code as specialized service even when packaged with shrink wrap and sold to millions. The extremes as represented by 'information wants to be free' at one end and UCITA at the other both negate concepts of code which do not meet the narrow mental models of the one who has framed the argument. History has shown that both speech without control and speech with excessive control yield effectively the same result - degradation of dialogue[40]. This proposal seeks to strike a balance in accordance with the traditions of intellectual property, the realities of economics, and the nature of code.

The current rubrics of intellectual property provide adequate flexibility to fit all those models, yet at this time the application of these models has no coherent underlying basis and is in such flux as to be reasonably said to be in disarray. A coherent application of intellectual property standards will enhance the Internet, allow multiple models of information exchange without placing the ideals of civil society at risk. The Digital Millennium Copyright Act is an excellent example of using a nuclear explosion when the hammer in the tool belt did not work[41] - when the task was best left to a screwdriver. Radical redesign of the toolbox is not needed. All that is necessary is a rational reconsideration of the tools and problems at hand.

Source code should be governed as speech because the core of source code is the communication of ideas. These ideas include the illustration how to solve a particular computing problem, the flow of information between user and code distributor, and the appropriate balance between speed, ease of use, and security. The distribution of source code is the exchange of ideas, ideas which are necessary for both democratic debate and scientific innovation.

The problem with the application of source code is the same as the problem with the application of ubiquitous contract law: one cannot expect any user to have the near infinite attention span necessary to examine the details of the code. A naive user loading source code an a computer would be unable to turn it into functioning code. Many a Linux user has found messages requiring extending the path or updating libraries are required to compile, and thus install, source code. Some such messages have turned prospective Linux users against the option; yet installation is far more simple thane examination of the body of code as a whole.

However, publishing the code would allow agents (in the traditional sense; for example, Consumer Reports) to examine code. It is the rare citizen who has read the law of the land, yet its openness serves us all (an analogy explored to its fullest in Lessig, 2000).

Object code should be treated as a machine because it is inherently functional. The design of licenses which prohibit reverse engineering illustrate that the distribution of the code is not intended to be a distribution of ideas, but a distribution of functionality42.

A distinction between these may be code written for an individual or unique application. In this case both the source code and the object code is provided. A strict application of the limits described here could prevent small companies making individually developed code from installing the code for clients. It is not unusual to make distinctions between services and products developed for the mass market and those which are customized for an individual user. An example of this which is close to home is the design of homes themselves. The governance of manufactured housing and stick built housing are fundamentally different. One is a code requiring particular implementations, for example, certain thickness of materials. In contract manufactured housing has to meet performance requirements. Similarly manufactured code and customer-produced code are distinct. Thus we propose governing such code as a professional service.

While this does not offer a complete detailed analysis of the regulatory framework, in this paper we have offered a practical and philosophically coherent model for the governance of code which will allow multiple markets and business models to thrive.

Footnotes

1. e.g., Johnson, D and Nissenbaum, H (1995),Computer Ethics and Social Value, Prentice Hall, N.J; Kling, R (1996), *Computerization and Controversy*, Academic Press, UK; Spinello, R and Tavani, H (2001), *Readings in Cyberethics*, Jones and Bartlett Publishers, January 2001.

2. e.g., DeBona, C, Ockman, S, and Stone, M (eds) (1999),*Open Codes: Voices From the Open Code Revolution*, O'Reilly, Cambridge; Lessig, L (1999), *Code and Other Laws of Cyberspace*, Basic Books.

3. e.g., Shapiro and Varian, H (1998)*; Information Rules*, Harvard Business School Press, Cambridge MA; Baldwin C and Clark, K (2000), *Design Rules*, MIT Press, Cambridge,

MA.

4. e.g., National Academy of Science (2000), *The Digital Dilemma: Intellectual Property in the Information Age*, National Academy Press, 2000; Branscomb, A (1995), *Who Owns Information*, Basic Books.

5. Eisenstein, E. L (1979), 'The Printing Press as an Agent of Change', Cambridge University Press, Cambridge, UK

6. Litman, J (1997), Reforming Information Law in Copyright's Image, , University of Dayton Law Review 587, (1997) 22: <http://www.msen.com/~litman/dayton .htm <http://www.msen.com/~litman/dayton.htm>>

7. National Academy of Science (2000), *The Digital Dilemma: Intellectual Property in the Information Age*, National Academy Press, 2000.
Similarly, the NCCUSL describes the threat faced by software producers as follows: Computer information is peculiarly vulnerable to dissipation of its value by copying. The genius of computers is their ability to retain and copy information. Copies of information look just like their originals. In fact, everything is a copy. There are no true originals. Copies can be duplicated in huge numbers and disseminated to millions of users in times measured in less than seconds. Therefore, those who invest capital, intellectual effort and labor into the creation of valuable computer information may lose the economic value of their products in seconds. Without the ability to control copying and dissemination of computer information, vendors risk losing everything. The risk is so great that without licensing, the development of computer information products could become uneconomical and the great economic benefit of computer information products could be lost. (NCCUSL UCITA summary:
<<http://www.nccusl.org/uniformact_summaries/uniformacts-s-ucita.htm>>.

8. Johnson-Laird, A (1997),'The Anatomy of the Internet Meets the Body of the Law', (1997) 22, University of Dayton Law Review. 465, (1997) 22
<<http://eon.law.harvard.edu/h2o/property/alternatives/johnson-laird.html>>.

9. Dyson, E (1995), 'Intellectual Value', Wired vol. 3.07, July 1995:
<http://www.wired.com/w ired/archive/3.07/dyson.html
<http://www.wired.com/wired/archive/3.07/dyson.html>>.

10. Barlow, J (1994),'The Economy of Ideas', Wired vol. 2.03, 1994:
<http://www. wired.com/wired/archive/2.03/economy.ideas_pr.html
<http://www.wired.com/wired/archive/2.03/economy.ideas_pr.html>>.

11. See e.g. Debian Project, 'What Does Free Mean?',
<<http://www.debian.org/intro/free>; Stallman, R (1996), Re-evaluating Copyright: The Public Must Prevail [Published in Oregon Law Review, Spring 1996]
<http://www.gnu.org/philosophy/reevaluating-copyright .es.html
<./vhttp://www.gnu.org/philosophy/reevaluating-copyright.es.html>> suggests that the DMCA altered copyright in exactly the wrong way; while traditional copyright stuck an

important but now inappropriate balance. The GPL depends on copyright.

12. SIIA web site: <<http://www.siia.net/piracy/programs/sftuse.htm>>

13. Note that the WIPO Copyright Treaty requires countries bound by the treaty to provide adequate legal protection and effective legal remedies against 'the circumvention of effective technological measures that are used by authors in connection with the exercise of their rights under this Treaty or the Berne Convention' (Article 11).

14. Fonda, D (1999),'Copyright Crusader', Boston Globe Magazine, August 29th, 1999: <http://www. boston.com/globe/magazine/8-29/featurestory1.shtml <http://www.boston.com/globe/magazine/8-29/featurestory1.shtml>>.

15. Some opponents were concerned with the process as well as the content. The Bill passed at the height of the impeachment crisis and '[t]here wasn't any debate,' said Eric Eldred, who later brought a constitutional challenge against the Act, 'no public consideration of the trade-offs being made. Where were the people charged with protecting the public domain - historians, archivists, free speech advocates? I was writing letters to newspapers, trying to get attention to the issue. But the public didn't realize what was going on; they didn't understand the consequences': Fonda, D (1999), 'Copyright Crusader', Boston Globe Magazine, August 29th, 1999: <http://www. boston.com/globe/magazine/8-29/featurestory1.shtml <http://www.boston.com/globe/magazine/8-29/featurestory1.shtml>>. See also Society of American Archivists, Text of Letter Sent on SAA Letterhead to Members of the Senate and House Judiciary Committees in Opposition to the Copyright Term Extension Act, November 1997:
< http://www.archivists. org/statements/copyextn.html <http://www.archivists.org/statements/copyextn.html>>.

16. Eldred v Reno, U.S. District Court, DC, Case No. 99-65, 1999.

17. Amazon's 1-Click patent #US05960411 can be viewed at: <http://www.patents.ibm.com/details?pn=US05960411 <http://www.patents.ibm.com/details?pn=US05960411>>

18. Tim O'Reilly of O'Reilly publishing reflected the views of many of those in the programming community when he made the following comments about the 1-Click patent: 'the Amazon 1-Click Patent is one more example of an 'intellectual property' milieu gone mad. In the first place, this patent should have never been allowed. It's a completely trivial application of cookies.... Like so many software patents, it is a land grab, an attempt to hoodwink a patent system that has not gotten up to speed on the state of the art in computer science. I'm not completely opposed to software patents, since there are some things that do in fact qualify as legitimate 'inventions', but when I see people patenting obvious ideas, ideas that are already in wide use, it makes my blood boil'. <http://www.oreilly.com/ ask_tim/amazon_patent.html <http://www.oreilly.com/ask_tim/amazon_patent.html>>.

19. League for Programming Freedom, 'Against Software Patents': <http://lpf.ai.mit.edu

/Patents/AgainstSP/asp-14.html <http://lpf.ai.mit.edu/Patents/AgainstSP/asp-14.html>>

20. League for Programming Freedom, 'Against Software Patents': <http://lpf.ai.mit.edu /Patents/AgainstSP/asp-05.html <http://lpf.ai.mit.edu/Patents/AgainstSP/asp-05.html>>

21. Lessig comments that 'An increasingly significant cost of Net startups involves both defensive and offensive lawyering - making sure you don't 'steal' someone else's 'idea' and quickly claiming as yours every 'idea' you can describe in a patent application.' (Lessig, L (1999), 'The Problem with Patents', The Standard, April 23rd, 1999: <<http://www.thestandard.com/article/display/0,1151,4296,00.html>>.

22. See also the League for Programming Freedom, 'Against Software Patents': <http://lpf.ai.mit.edu /Patents/AgainstSP/asp-07.html <http://lpf.ai.mit.edu/Patents/AgainstSP/asp-07.html>>.

23. League for Programming Freedom, 'Against Software Patents': <http://lpf.ai .mit.edu/Patents/against-software-patents.html <http://lpf.ai.mit.edu/Patents/against-software-patents.html>>.

24. Tim O'Reilly, in his email dialogue with Richard Stallman, February-March 2000: <http://w ww.oreillynet.com/pub/a/patents/2000/03/09/stallman.html <http://www.oreillynet.com/pub/a/patents/2000/03/09/stallman.html>>.

25. The Economist notes that: ' Increasingly, companies realize that among the few remaining barriers to entry are the ones that the government hands out in the form of 20-year monopolies' ('Patent Wars: Better get yourself armed. Everybody else is', The Economist, April 8th, 2000). It adds: ' IBM is now getting ten new patents every working day'.

26. Notice I refer to kinds of code since type and class both have specific meanings in the context of software.

27. Ullman, E (1995), 'The Dumbing-Down of Programming' Salon.com, 21 March 1995: <http://www.salon.com/21st/feature/1998/05/cov_12feature.html>.

28. CGI stands for common gateway interface. Perl, Javascript and CGI are the most widely used scripting languages

29. Greenia, M W (2001), *History of Computing: An Encyclopedia of the People and Machines that Made Computer History,* (January 2001) Lexikon Services (UK) 2001. Grace Hopper also created the first high-level language, COBOL.

30. Junger v. Daley, No 96-CV-1723 (N.D. Ohio, July 2, 1998).

31. Bernstein v. US Department of State, 945 F. Supp. 1279 (ND Cal. 1996).

32. Karn v. United States Dep't of State, 920 F. Supp. 1, 9 n.19 (D. D.C. 1996).

33. United States Court of Appeals Decision, Junger v Daley, April 4th 2000, <http://samsara.law.cwru.edu/victo ry.txt <http://samsara.law.cwru.edu/victory.txt>>.

34. Perry Barlow, J (1994), The Economy of Ideas, Wired 1994, vol. 2.03: <http://www. wired.com/wired/archive/2.03/economy.ideas_pr.html <http://www.wired.com/wired/archive/2.03/economy.ideas_pr.html>>.

35. Schneier, B (1995), *Applied Cryptography*, John Wiley & Sons, NY, NY.

36. see, for example, Froomkin A M (1995), 'The Metaphor is the Key: Cryptography, the Clipper Chip, and the Constitution', 143 U. Penn. L. Rev. 709 (1995). <http://www.law.mi ami.edu/~froomkin/articles/clipper.htm <http://www.law.miami.edu/ ~froomkin/articles/clipper.htm>>.

37. Zittrain, J (1999) 'The Un-Microsoft Un-Remedy: Law Can Prevent the Problem That it Can't Patch Later,' 31 Connecticut L. Rev. 1361 (1999). <http://papers.ssrn.com/sol3/pre_papers.cfm?ABSTRACT_ID=174110 <./vhttp://papers.ssrn.com/sol3/pre_papers.cfm?ABSTRACT_ID=174110>>.

38. See, for e.g., R. Stallman,Patent Reform is Not Enough: <http:// www.gnu.org/philosophy/patent-reform-is-not-enough.es.html <http://www.gnu.org/philosophy/patent-reform-is-not-enough.es.html>>.

39. We mean free as in 'free beer', not free as in 'free speech'. As Stallman notes, 'Free software is a matter of liberty, not price. To understand the concept, you should think of 'free speech', not 'free beer''. See: <http://www.fsf.org/philosop hy/free-sw.html <./vhttp://www.fsf.org/philosophy/free-sw.html>>.
In contrast, information wants to be free argues that all information should have zero price. While information wanting to be free is sometimes confused with free software, these are fundamentally different uses of the word.

40. Darnton, R, (1985), *Literary Underground of the Old Regime,* Harvard University Press Cambridge, MA.

41. Lutzker, A P (1999), Primer on the Digital Millennium: What The Digital Millennium Copyright Act and the Copyright Term Extension Act Mean for the Library Community, March 1999: <http://www.ala.org/washoff/primer. html <http://www.ala.org/washoff/primer.html>>.

42. Most proprietary licenses prohibit reverse engineering. The DMCA prohibits reverse engineering for encryption software, creating criminal penalties as well as a private right of action.

This proposal was supported in part by NSF CAREER grant 9985433, and an equipment grant from HP. I would I would like to acknowledge J. Cohen and S. Garfinkle for their

thoughts.