Executable Code is Not the Proper Subject of Copyright Law

A retrospective criticism of technical and legal naivete in the *Apple V. Franklin* case

Matthew M. Swann,
Clark S. Turner, Ph.D.,
Department of Computer Science
Cal Poly State University
November 18, 2004

Abstract: Copyright was created by government for a purpose. Its purpose was to be an incentive to produce and disseminate new and useful knowledge to society. Source code is written to express its underlying ideas and is clearly included as a copyrightable artifact. However, since *Apple v. Franklin*, copyright has been extended to protect an opaque software executable that does not express its underlying ideas. Common commercial practice involves keeping the source code secret, hiding any innovative ideas expressed there, while copyrighting the executable, where the underlying ideas are not exposed. By examining copyright's historical heritage we can determine whether software copyright for an opaque artifact upholds the bargain between authors and society as intended by our Founding Fathers. This paper first describes the origins of copyright, the nature of software, and the unique problems involved. It then determines whether current copyright protection for the opaque executable realizes the economic model underpinning copyright law. Having found the current legal interpretation insufficient to protect software without compromising its principles, we suggest new legislation which would respect the philosophy on which copyright in this nation was founded.

Table of Contents

Introduction

Copyright is an artificial legal concept created for the purpose of encouraging innovation in the sciences and the arts, impacting not only professionals and scientists but also society as a whole. Thanks to the utilitarian nature of copyright as established in the Constitution, society benefits from the discoveries of the learned while those on the cutting edge of innovation are allowed to profit from the publication of their research. This tradeoff, termed the social bargain, is the underpinning of the modern interpretation of copyright.

Traditionally thought only to apply to literature and the arts, copyright has in recent years been extended to cover software as well. Crucial to this extension of the law is determining whether it continues to respect the bargain forged by our Founding Fathers between society and inventors—any new application of copyright should hold true to the spirit in which it was established. By examining the context in which copyright in this nation was born, we can divine the responsibilities that copyright holders have in holding up their end of the social bargain.

In the following pages we will examine copyright's origin and development, then look at software and analyze the characteristics which set it apart from traditionally copyrighted media. Finally, we will look at copyright as it applies to software to determine whether the current state of affairs reflects the incentives which copyright was created to provide. Having covered this ground, we can look critically at how well copyright is doing that job and whether there are other alternatives to be considered.

The Origin of Copyright

The Idea is Born

Copyright began in fifteenth century England with the establishment of the Stationers' Guild. Formed in 1403 by bookbinders, scriveners, and stationers to protect their trade from outside competition, it was granted an official charter in 1557 by Queen Mary Tudor[1]. This charter gave the Guild a monopoly over the printing and publishing trade— new books first had to be licensed by the Crown, then entered into the Stationers' register; the first stationer to enter a book into the register then owned an exclusive (but transferable) right to its duplication. Under this system both the Crown and the Stationers' Guild benefited from the monopoly. The Crown used its licensing requirement to suppress the publication of books that contained "dangerous" ideas, while the guild enjoyed a virtual monopoly on the book trade. England's Long Parliament further strengthened this policy during the mid-seventeenth century, as new laws were added which threatened "fines and imprisonment of authors, publishers, sellers, and buyers of scandalous or libelous materials"[2]. By the end of the seventeenth century, presses outside of London, Oxford, and Cambridge were banned as Parliament took tighter and tighter control of printed works.

[1] See also the Star Chamber Decree of 1586

[2] Harry Hillman Chartrand. "Copyright C.P.U. : Creators, Proprietors, and Users." *Journal of Arts Management, Law & Society* Vol. 30, No. 3, Fall 2000

Copyright in the seventeenth century and before was based on the *natural rights* model, where the right of the author of a work to control its publication and dissemination was taken to be naturally endowed by human nature. (This is in contrast to the *utilitarian* model, where copyright is a right conferred by law with the purpose of benefiting society as a whole.) Rather than holding on to their rights, however, authors were forced to sell their rights to the Stationers' Guild for a one-time fee upon the initial publication of their work, rescinding any rights to subsequent royalties[3]. The protections afforded by copyright, then, served only the interest of the Crown and the publishers themselves. Furthermore, the Guild offered their services without regard to any guiding principle save self-preservation:

> *[Queen] Mary incorporated the Stationers' Company "to set up a mode of regulating the English printing trade that would facilitate the efforts of the Romish clergy to stamp out the Protestant Reformation." But the motives of the stationers "were of a less exalted kind." Thus, Elizabeth, relying on the stationers' self-interest, confirmed the Charter to turn the stationers to support the English, rather than the Romish church, and the Stationers' Company became, in turn, the instrument of the Stuarts against the Puritans, in the early seventeenth century; the instrument of the Puritans, against their royalist enemies, when the Puritans came to power; the instrument of the royalists against the Puritans, after the Restoration; and, for a brief time, the instrument of the triumphant Whigs, after "the glorious Revolution," of 1688. But through all these vicissitudes, the stationers themselves steadfastly remained, what they had always been, eminently practical men; and they consistently protected their monopoly[4].*

While this system remained in place for the next 150 years, there grew among members of English society and of the Parliament a sentiment that publication should be free from suppression and censorship[5]. However, under the natural rights model set forth by Queen Mary and the Guild, such an idea would never be able to take fruit.

A New Beginning

Copyright's modern interpretation took on its first form in 1710 with the Statute of Anne[6]. Considered to be the "turning point in the history of copyright", it established the modern meaning of copyright and was the basis for the Constitutional definition of copyright as set forth by the Founding Fathers. It also signaled a shift from the natural rights model of copyright to one based on utilitarian principles. Among the reforms instituted in the Statute[7]:

- It acted as an incentive to "[encourage] learned men to compose and write useful books." By assigning rights to the author of a work rather than the publisher, the Statute provided a financial incentive to those who innovate and publish their discoveries. No longer was publishing a work tantamount to giving up one's rights to it; an author was free to publish his work without forfeiting his rights.

[3] Eisenstein, Elizabeth L. The Printing Revolution in Early Modern Europe 83-84 (1983)
[4] Patterson, L.R. "Copyright and 'The Exclusive Right' of Authors." *Journal of Intellectual Property* Vol 1 No 1 Fall 1993
[5] Keith Aoki. *Authors, Inventors, and Trademark Owners: Private Intellectual Property & the Public Domain*, 18 Colum-VLA J.L. & Arts 1 (Part I) & 191(Part II) (1993-94).
[6] Copyright C.P.U
[7] The Statute of Anne, 1710

- It contributed to public knowledge. To this end, the Statute dictated that copies of every newly published work must be deposited in libraries around the country in order that the public may read and learn from them. It also implemented a system to address grievances about overpricing, in order that the public might have a recourse against those who wish their works to not be readily available.
- It established a mechanism for copyrighted books to fall into the public domain: books which had previously been copyrighted fell into the hands of the public within twenty-one years, and newly copyrighted books were limited to a lifespan of fourteen years with one fourteen year renewal. In doing so, Queen Anne abolished the perpetual rights enjoyed by the Stationers in the century and a half prior to the Statute.
- Finally, the Statute limited the rights of copyright holders to the printing and reprinting of works— the right to use the information contained in the works was the property of society at large.

Rather than portraying copyright as an inalienable human right, the Statute of Anne established a utilitarian system whose very nature was designed to benefit the good of society. By stimulating authors and scientists to publish their work and retain their rights after doing so, Anne ensured that society as a whole would be the ones to gain from the knowledge of the few.

The Social Bargain

As well as changing the legal tenets of copyright, the Statute changed the underlying economic model as well. Before 1710, copyright legislation implemented a system of control and regulation; the Statute of Anne established a give-and-take relationship between authors and society which is termed the *social bargain.* In this system, authors are granted a limited monopoly on the expression of their ideas. Society, on the other hand, gains from publication of these ideas, where they are free to be used at will. Note that the utilitarian model of copyright is only concerned with an author's particular expression, not the ideas he is communicating through his work. When a brilliant mathematician publishes a volume of knowledge, the ideas and concepts contained in the volume immediately become a part of the public domain— the precise manner in which he expressed his ideas, however, is still owned by the author himself. Although even the author's expression lapses into the public domain after a certain amount of time, he retains the rights to it long enough to profit from his work.

An important footnote to the social bargain is that it's crucial that the valuable ideas in the work, be they literary, mathematical, or scientific, are disclosed. Owning the copyright to a work that is never released publicly benefits no one— if I scribble on a napkin then tuck it into my pocket and claim copyright on it, I've neither added nor taken away from the public's collective knowledge. The Statue of Anne's intent was to *prevent* brilliant authors from doing exactly that; since publishing their works prior to the Statue meant giving up their rights as authors and owners of their works, they often chose instead to keep their works hidden away. The Statue changed that behavior not only by allowing authors to retain ownership of the works they published, but also by dictating that all published works be deposited in libraries for public consumption. Similarly, it stands to reason that for copyright to be truly a "bargain" between authors and society, there has to be some value imparted to society by an author's expression of his work. It makes little sense to grant a monopoly on a work's publication when the work has no value.

These two ideas form a convenient test by which we can evaluate those who would wish to protect their works by copyright:

- The work must be published. If copyright exists to increase public knowledge, then my work must be available for the public to see and learn from.
- The work must be of value, whether that be of an artistic or intellectual nature. If I am enjoying a monopoly by claiming the copyright on my work, I should be giving something useful back to society in return.

It stands to reason that if either of those tests fail to apply to a work for which copyright protection is being evaluated, then copyright is not the proper venue with which to protect it. Authors who wish to claim copyright protection for their works should see to it that their actions are in the spirit of the social bargain.

Copyright and the Constitution

At the time the Founding Fathers sat down to define copyright in this country, the Statute of Anne was the current basis for such legislation. The fact that copyright required the creation of a monopoly (which the Founding Fathers sought to avoid as much as possible) presented some difficulty, but it was judged that the benefit to the public of such legislation would far outweigh any disadvantage it may pose. In the *Federalist Papers*, James Madison wrote:

> The utility of this power will scarcely be questioned. The copyright of authors has been solemnly adjudged, in Great Britain, to be a right of common law. The right to useful inventions seems with equal reason to belong to the inventors. The public good fully coincides in both cases with the claims of individuals. The States cannot separately make effectual provision for either of the cases, and most of them have anticipated the decision of this point, by laws passed at the instance of Congress[8].

Madison's statement tells us several things about copyright at that time. First, copyright was a right recognized by common law, not a man-made and artificial right. Second, it was acknowledged that the author's monopoly under the Statue of Anne provided an effective incentive to create new works. Third, Madison acknowledged that the public interest coincides *fully* with the interest of authors to obtain exclusive rights to their works[9]. The effectiveness of the Statue of Anne and the corresponding social bargain thus provided ample ground on which to build the powers of copyright into the United States Constitution.

Section 8.8 of the United States Constitution paraphrases the Statute when it states that Congress has the power to "promote the progress of science and the useful arts, by securing for limited times to authors... the exclusive right to their respective writings"[10]. Like the Statute of Anne before it, the Constitution's definition of copyright is strictly utilitarian. Rather than assigning rights to publishers, it assigns them to authors; rather than being solely for the benefit of the rights holder, copyright's purpose is first and foremost to benefit the public by advancing the

[8] The Federalist No. 43 at 270-271 (Rossiter ed. 1961).
[9] Patry, William F. "Copyright Law and Practice." The Bureau of National Affairs, Inc. 2000.
[10] The United States Constitution art I § 8

state of science and art in society as a whole. Although case law and legislative measures have since expounded on the definition of "authors", "writings", and "limited times" to suit technological advances and corporate whims, the concept remains the same today as it did some two hundred years prior.

The Basics of Software

Software's Dual Nature

Although one could argue that copyright has successfully kept up with new forms of artistic and scientific expression, software's unique nature has caused its share of problems. At first glance, it seems a world apart from the literary works which copyright was first applied to. This is only partly true; while software has many attributes that make it a unique form of expression, it shares key attributes with literary works. Crucial to the understanding of software as it fits into the world of copyright is its dual nature— in contrast to a book, which is written and released to the public in the same form, software exists in one form when it is being written and is published in an entirely different form.

Source Code: Software in Human-Readable Form

Software begins its life as a compilation of text composed of special symbols and terms arranged in a highly specific grammar; easily written, read, and revised by software engineers, this text is called *source code*[11]. Source code is where the knowledge and ideas communicated by software engineers is kept—by writing source code, a software engineer tells the computer how to perform a specific function. In doing so, he uses design patterns and practices which he knows to be effective at solving the problem at hand. Likewise, one software engineer can read the source code of another and learn from the patterns and concepts used. This flow of ideas between software engineers allows us to make the argument that source code is a literary work.

It is well-known that specialists in a field communicate with each other using their own specific language, unique to their area of specialization[12]. Physicists, for example, have their own terminology to describe the arena in which they work. To communicate new ideas, they write research papers and publish them in a journal read by their peers. Like a research article, source code communicates the concepts and discoveries of the software engineers who authored it. In the same way that civil engineers design a bridge, software engineers rely upon a wealth of design patterns which are known to be flexible and robust. When new design patterns and software algorithms are discovered, they cannot effectively be communicated to other software engineers in any way besides source code. For this reason, it can be said that the primary language of software engineers is source code. In the same way that mathematicians share ideas by using equations and symbols, software engineers communicate with each other in source code. Source code can thus be viewed as a human-readable literary work which expresses the ideas of the software engineers who authored it.

[11] Samuelson, Pamela. "Contu Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form." *Duke Law Journal* Vol 1984:663. pg 683.
[12] A brief search of the internet or a local bookstore reveals dozens of dictionaries devoted solely to technical terms.

Object Code: Software in Machine-Readable Form

Although source code allows humans to communicate directly with each other, computers cannot directly make use of software in this form. Before source code can be anything besides a special language for software engineers to communicate with one another, it must be *compiled* into a format that can be directly executed by the computer. This format, called *object code*, is written in the language of the computer itself[13]. Because it is written in a machine-readable format, it is extremely difficult, costly, and time-consuming to read—even for experts in the field[14]. As a result, once the translation has been made, there's no going back—in order to extract useful information about the structure and overall design of a computer program, one must have access to the source code. However, object code is what makes a computer program function, so that is what is distributed to the public in the form of retail software.

If source code is a literary work written by and for software engineers, then the conversion from source to object code is a lossy translation of the original work, one in which the part of the work addressed to humans is discarded in order for the resulting data to be understood by a machine audience. To some, this is identical to translating a work written in English to a different language. If this is indeed the case with object code, then the resulting machine-readable code is still a literary work, albeit one stripped of many of its concepts and design decisions.

The case can also be made that object code, once compiled from source code and translated into the language of the computer hardware, has more in common with hardware than with any literary work. Since any algorithm could be created exclusively in hardware, object code is simply a replacement for that dedicated hardware[15]. The function of a compiler, then, is to translate human logic in the form of source code into pure function. For example, using a word processing application requires no knowledge of the source code that brought the application to life; it is the object code you brought home on CD that gives your computer function. This blurs the line between software and the function it was intended to perform, which has important repercussions when we try to apply copyright to object code later on.

Copyright Protection for Software

Software's Status Under Copyright

As can be inferred from the context of this work, software is covered by copyright. Although the Copyright Act of 1976 defined software as a literary (and thus copyrightable) work, it painted the definition in broad enough strokes that it has been up to case law to determine what exactly copyright applies to[16]. For instance, it is no stretch to see how copyright protection can apply to source code. In 1983, however, *Apple v. Franklin* made it clear that both source *and object code* were copyrightable[17]. Simply because legislation states that a work is copyrightable does not mean that software developers must take advantage of that fact, though— software's unique nature affects how businesses choose to protect their intellectual property.

[13] Samuelson, "Contu Revisited" pg 686
[14] Note that the DMCA and most EULAs have made this illegal despite the effort involved
[15] Samuelson, "Contu Revisited" pg 673, 680
[16] The United States Copyright Act of 1976
[17] Apple Computer, Inc. v. Franklin Computer Corp.

Copyright and Source Code

Source code's human-readable nature presents a particular difficulty to businesses. Software engineers work for months or even years to develop new software; crucial to this development process is the discovery and implementation of new algorithms and design patterns. These innovations are the heart of the software, enabling a software company to maintain a competitive advantage over others in the same market who may not have discovered how to perform the same functions. It is critical, then, that these new discoveries not be released to the public. Because copyright protects only the expression (the source code) and not the underlying ideas (the algorithms) in a work, it affords no protection to the innovations which software engineers have invested time and money in discovering. Instead, they opt to protect their source code under state trade secret laws, which require no disclosure and afford protection from disclosure of both the expression and ideas in source code.

Copyright is not completely swept under the rug when a software developer wishes to protect his source code, however. While holding the copyright to his source code may not afford a developer any rights while it remains unpublished, it can be helpful in matters of contract disputes or licensing infringements where business partners may have wrongfully used portions of the developer's source code. If a trade secret should fall into the hands of the public (by a business partner's carelessness, for example), all protections afforded by trade secret law disappear. On the other hand, the developer whose source code was improperly distributed can still make a copyright infringement claim to regain control over the publication of his work[18].

Copyright and Object Code

Although a developer may choose to restrict the dissemination of his product's source code, his work is useless unless the corresponding object code is released to the public. When a developer releases his object code to the market in the form of a finished application, he often chooses to claim the copyright on his work. Copyright restrictions on object code are used by developers in order to keep others from stealing all or part of their code and releasing it as an independent work; rather than appeal to misappropriation laws (which may vary from state to state), developers have found that the legal rights conferred upon them by copyright afford all the protection they need[19]. Although copyright's protection is designed to only last for a "limited time"[20], it has been extended to offer up to 95 years of protection, effectively preventing software from ever lapsing into the public domain. Aside from copyright protection itself, software developers also benefit from the obscurity that object code lends to its underlying ideas—reverse engineering to get back to the original source code is so difficult as to be commonly held as impossible[21].

The incentive to profit from the distribution of one's object code also provides an incentive to keep source code protected as a trade secret. If the source code to an application became public, skilled individuals could reproduce the application illegally at zero cost by simply compiling the

[18] See SCO v. IBM for an example
[19] Apple Computer, Inc. v. Franklin Computer Corp.
[20] The United States Constitution art I § 8
[21] Since most EULAs make reverse engineering illegal, this effectively means that the purchaser of a software product is barred by law from reading what he purchased. How's that for a strange turn of events?

source code into object code. Thus, in order for object code to have value to consumers, the corresponding source code must not be available. Software developers are aware that the object code they release will still be duplicated illegally, but they are willing to mitigate their losses by allowing the expression of their work to be duplicated while their ideas stay undisclosed.

	Literature		*Software*	
	Novel	**Research Paper**	**Source Code**	**Object Code**
Author	Public	Professionals	Software Engineers	N/A (Compiled)
Audience	Public	Professionals	Software Engineers	Machines
Copyrightable?	Yes	Yes	Yes	Yes
Ideas Disclosed?	Yes	Yes	Yes	No
Human Readable?	Yes	Yes	Yes	No[22]
Publication Status	Public	Research Journals	N/A (Trade Secret)	Retail Software

Table 1. A summary of copyright as it applies to literature and to software.

Protections Offered by Software Copyright

For companies investing heavily in software research and development in the hopes of gaining a competitive advantage over others in their market, copyright affords nearly perfect protection against losing their advantage. Not only does copyright allow them to publicly release their object code without the fruits of their research being revealed, it lasts far beyond the time at which their software ceases to provide any new income. Additionally, it offers them recourse should they lose their source code's status as a trade secret.

Because businesses can release software in object code form without divulging the innovation that lies behind it, software copyright gives them protection on the expression of their ideas while simultaneously providing strong incentives to keep those ideas hidden. Even patent law, considered to be the most powerful of intellectual property protections, demands full and exhaustive disclosure of the invention being claimed before protection is given. By giving software companies strong and lasting protection while mandating no disclosure due to software's unique nature, software copyright has become more powerful than patent protection while removing the attributes of each which would benefit the public and maintain a competitive balance in the market.

Software and the Social Bargain

The Spirit and the Letter of the Law

Although claiming the copyright on a work may give an author significant legal protection, copyright carries with it certain ethical responsibilities as enumerated earlier in the social bargain. For copyright to be in the spirit of the social bargain, an author has a responsibility to publish the copyrighted work and the responsibility to see to it that the work is of some value. Therefore, protection which is legal under current legislation may be in direct contradiction to the spirit, rather than the letter, of the law. In this section we will examine whether the protection afforded to software meets these criteria.

[22] Only with much difficulty

Source Code

As discussed earlier, software companies commonly withhold their source code from publication, seeking instead to protect it as a trade secret. Although they copyright their work, they rarely claim that right unless their work is inadvertently disclosed. Our first test of whether a work is subject to copyright, however, is whether or not it has been published; as shown earlier, the utilitarian nature of copyright makes it nonsensical to copyright a work which remains secret. Thus, software companies who hold their source code under trade secret law fail our first test.

Since source code remains undisclosed at most software companies, it stands to reason that they are not giving anything back to society by way of their copyright. To further the software engineering profession, engineers would need to share their source code so that innovations could be shared among professionals in the field. Since this does not happen, it is clear that any copyright applied to undisclosed source code does not meet the original purpose of copyright legislation. Thus, the argument can be made that source code as commonly held by software companies today should not qualify for copyright protection, even if inadvertently released. Although this removes a legal tool by which companies can attempt to regain control of their source code, we will present a solution later in the paper.

On the other hand, there are definite cases where source code could rightly be copyrighted. If the source code were to be made available to other software engineers, this would satisfy both the publication and social bargain aspects of copyright. In fact, this is the tactic taken by members of the open source development community. Rather than writing software and holding the source code privately, both source and object code are released publicly under licenses such as the GPL. By doing so, developers add to the wealth of knowledge shared by other software engineers and enable their work to be updated, fixed, and built off of by others within the open source community.

Object Code

Unlike source code, object code is distributed to the public at large in the form of retail software. When a person purchases a copy of Microsoft Office, he or she brings home a CD with the applications in object code form. Thus, its wide distribution ensures that object code passes our first test. When it comes to giving back to the community and upholding the social bargain aspect of copyright, however, object code presents a much more interesting case. Crucial to the issue is what is being given back to the community when object code is distributed—there are two differing interpretations, each of which benefits one particular group of people who are concerned with what copyright might mean.

From the perspective of software companies whose primary source of revenue is the creation and distribution of retail software (and thus object code), the benefit their software gives to society is in the form of functionality for the end user's computer[23]. That is to say, in return for a limited monopoly on the object code they sell, software companies give consumers functionality they did not have before. In the case of Microsoft Office, this functionality comes in the form of a word processor, spreadsheet, presentation application, and many other components. In this interpretation, the audience who receives the benefit from copyright is the general public; anyone

[23] Samuelson, "Contu Revisited" pg 737

who purchases and installs Office reaps the benefits of the software. However, this line of argument also disqualifies copyright from protecting object code; if object code is merely functionality, then patent, not copyright, is the branch of intellectual property law designed to protect it.

Software companies are not the only ones concerned with copyright, however. Software engineers are concerned with advancing the state of their profession and creating products with a high level of quality; crucial to this noble quest is the ability to learn from the work of other engineers[24]. Like the civil engineering analogy made earlier, software engineers must work together to contribute new and tested ideas to the field in order that future engineers may create better designs. From the perspective of a software engineer who strives to learn from cutting-edge practices, the benefit from the copyright on object code is an increase in their knowledge and in the design patterns they have available in their work. However, this is where software engineers find fault with object code's participation in the social bargain. Far from communicating ideas in a human-readable format (as source code is designed to do), object code must be painstakingly reverse-engineered to extract what few design ideas are still present after being converted from source code[25]. Furthermore, most end user licensing agreements make reverse-engineering illegal[26]! Software engineers interpret the audience of the benefit provided by copyright in the social bargain as being other software engineers; by releasing object code under copyright, the engineers who created the code should be making concepts and ideas available to their colleagues for the purposes of advancing their field.

The contradiction between these two interpretations are clear. If (as software development houses contend) object code is merely functionality, then copyright protection is inappropriate; patent, not copyright, was designed to protect an element's function[27]. On the other hand, if object code is merely a translation of source code, then it should give its innovations and ideas back to the software engineering community. As noted above, object code fails to do so in every way.

Conclusions

This dichotomy where each interpretation proves itself to be unsuitable for copyright protections is the ultimate problem with software's entry into the intellectual property scene: although it can be said to resemble other forms of expression, no current protection fits it while preserving the ideological and practical purposes for which the protection was created. The social bargain in particular breaks down when it meets software; in order to grant a temporary monopoly to software publishers, its source code must be kept secret, but in order to give ideas and concepts back to the software engineering community, source code must be disclosed. To apply copyright protection to software thus means giving up the innate requirement of publication as well as its founding principle of advancing the state of the sciences. As if this weren't enough, using copyright to protect object code as a functional product blurs the lines between copyright and patent protection. Faced with this dilemma, the field of intellectual property law can respond in one of two ways: either reshape copyright to protect software in all its uniqueness, or draft

[24] ACM Code of Ethics § 2.1
[25] Samuelson, "Contu Revisited" pg 683
[26] An example can be found at http://www.microsoft.com/whdc/devtools/IFSkit/ServerIFSEULA.mspx
[27] See the United States Patent and Trademark Office at http://www.uspto.gov/

legislation to establish a new field of law which covers software and is dedicated solely to managing the interests of software engineers and companies who depend on software and innovation to provide revenue.

New Directions

Protecting Software's Unique Form of Expression

It is important to note that this issue of how copyright should apply to software is hardly new; in "CONTU Revisited", Dr. Pamela Samuelson argued against copyright protection for object code, opting instead for new intellectual property law which would include federal misappropriation legislation in order to handle cases such as *Apple v. Franklin* without stretching copyright beyond what it was originally intended for[28]. Although this successfully protects the interests of software corporations when releasing their object code for distribution, it doesn't address the role of source code in the field of intellectual property law. Ideally, the property law that protects source code should have the same utilitarian purposes as copyright law was designed to have: it should protect the business interests of those who have invested in innovation, while allowing the public (and specifically software engineers) to reap the benefits of these new concepts. As we have seen, software engineers need access to source code in order to learn these new concepts and profit (in a knowledge-based sense) from their innovation. Thus, an ideal proposal for protecting source code should balance the need for its disclosure as well as the need to recoup the investment made in its creation.

To that end, I propose that a new area of intellectual property legislation be created specifically to cover software. As suggested by Dr. Samuelson, it should cover object code only inasmuch as it needs to be protected from misappropriation. However, for any object code which is distributed:

- The corresponding source code must be deposited in the Library of Congress.
- The source code will remain undisclosed for a period of time during which its creator has an opportunity to recoup the expenses incurred in the design and development of the product. I suggest three to four years as adequate time for this to occur[29].
- After this period of time, the source code will be made available free of charge (or at a nominal fee for the purpose of supporting the infrastructure needed to make this happen) to any and all interested parties.

Once the source code is freely available, it makes little sense to legislate any protection on it. (Whether misappropriation protection would be logical is largely a matter of determining what constitutes theft of the original code. Make the definition too narrow, and it does nothing; make it too wide, and the source code will serve no purpose except to stifle the use of its design and ideas.)

[28] Samuelson, "Contu Revisited" pg 768
[29] Samuelson, "Contu Revisited" pg 766

	Old Regime	New Regime
Monopoly Duration	95 years	3-4 years
Disclosure of Ideas	None	After 3-4 years
Disclosure Method	Object code[30]	Source code
Societal Benefit	Functionality	Functionality and ideas (source code)
Interoperability	Discouraged through nondisclosure	Encouraged via open source code
Security Incentive	Security through obscurity	Well-written source code
Public Domain Concept	Non-existent	Secured through source code release

Table 2. Terms of copyright's social bargain before and after proposed legislation.

Effects

Such legislation could be expected to have several effects.

- It would encourage interoperability among competing software publishers by allowing each to inspect the file formats of the other.
- It would present a strong challenge to software developers who contend that security through obscurity is good enough to protect the users of their applications. Since their source code will be made available to anyone and everyone, regardless of intention, developers would have motivation to write clear, secure, and verifiable code that will withstand public inspection.
- It would aid software developers in fixing bugs that were not caught in the testing process; interested parties would have the opportunity to lend their debugging skills once the source code became public.
- It would allow third parties to take up development of software whose developers have lost interest in supporting or improving it.
- It would protect the expression of an idea (in the spirit of copyright) rather than protecting functionality, which is the role of patent
- It would cement the concept of public domain for software when currently no mechanism exists to do so.
- It would allow software developers to learn from one another's work; no longer would any corporation or group of software engineers have a monopoly on their innovations.

As the spirit of copyright intended, software engineers receive a limited monopoly on the publication of their work; after a period of time, the public would have unfettered access to the ideas contained within the work. Unlike copyright, however, both cannot occur at the same time.

The tradeoff inherent in this legislation is that it would expose proprietary file formats and functionality which software developers have long kept secret. To many software developers, this proprietary information is the heart and soul of their competitive advantage—if the majority of the market uses one company's proprietary file format, the company can tout it as a de facto standard without making it open to other developers. On the other hand, if the company's source code were made available after a period of time, third-party developers could achieve perfect compatibility with the standard. Open access to source code could also help third-party software developers maintain feature parity with well-established competitors in the market. Although this disclosure of features and formats could be viewed as a detriment by well-established

[30] EULA restrictions on reverse engineering render this ineffectual.

software companies, it may act instead as an incentive to introduce new features and capabilities in order to stay one step ahead of their competitors. Since their source code remains undisclosed for a period of years after their application reaches the market, it seems feasible for companies to have introduced new software by that time. In light of the harm many would claim such legislation would cause, it appears that the benefits both to the consumer as well as to the field of software engineering far outweigh any disadvantages that individual players in the software industry would suffer.

Moving Forward

When the Statute of Anne demolished the natural rights model of copyright to establish one based on the social bargain, it set a precedent for the understanding of copyright not only in Europe but here in America. Our Founding Fathers designed copyright into the Constitution with the intent of promoting the free exchange of knowledge, balancing the needs of the author with the public good. Unfortunately, software copyright has upset that balance by meeting the needs of intellectual property creators while ignoring the good of science and of society as a whole. When compared with the behavior it was designed to promote, it is clear that copyright's relationship with software is deeply and fundamentally flawed. New legislation is needed not only to recognize software's uniqueness but also to reflect the social bargain that our Founding Fathers saw the need to establish. By allowing software publishers to recoup the expenses incurred in developing new software while simultaneously driving innovation by sharing ideas among software professionals, the legislation described in this paper would hold true to the social bargain in a manner currently unseen by copyright in its current form.

About the Authors

Matthew Swann is an undergraduate senior in the Computer Science department at California State Polytechnic University, San Luis Obispo. His interests include human-computer interaction, object-oriented design, copyright, and software testing. This paper was written under the kind and insightful guidance of Dr. Clark Turner, Professor of Computer Science at Cal Poly.

Clark S. Turner is an Associate Professor of Computer Science at the California State Polytechnic University in San Luis Obispo. He is also an attorney with interests in software IP and liability issues.