INTO THE TWENTY FIRST CENTURY WITH A TWENTIETH CENTURY SCHEME FOR INTELLECTUAL PROPERTY PROTECTION FOR COMPUTER SOFTWARE: A NEW PROPOSAL FOR SUI GENERIS LEGISLATION IN LIGHT OF RECENT CASES GRANTING BROAD COPYRIGHT PROTECTION FOR USER INTERFACE ASPECTS OF PROGRAMS AND THE PATENT OFFICE'S CONSISTENT DIFFICULTIES EVALUATING SOFTWARE APPLICATIONS

Daniel S. Kirshner
Candidate for Master of Intellectual Property
Franklin Pierce Law Center
May 3, 1994

FRANKLIN PIERCE
LAW CENTER LIBRARY
CONCORD, N. H.
JUL 18 1997

INTRODUCTION

Recent events in the law of intellectual property protection for computer software dictate that the time has come for Congress to seriously consider a new sui generis type of protection. In the world of copyright protection for computer software, the courts have consistently had difficulty defining the scope of protection for non-literal elements, and the recent decision in Lotus Development Corporation v. Paperback Software granting broad copyright protection to user interface portions of software illustrates this point. [1] In the world of software patents, the United States Patent Office has consistently had difficulty evaluating patent applications relating to software inventions, and two events within the past several months - a broad patent issued to Comptons New Media and a federal jury verdict finding infringement of a broad patent issued to Stac Electronics - illustrate this point.

These cases illustrate some of the problems surrounding the present scheme of intellectual property protection for computer software. This paper advocates a sui generis form of intellectual property protection for software - a whole new body of federal law regulated by an agency of the federal government, either acting independently or as a branch of the Patent Office. This paper will examine the problems with the scheme for intellectual property protection for software as it exists now, will examine various alternatives, and will conclude

that the best alternative is a sui generis form of legislation. This paper will then set forth some specific recommendations regarding the parameters of the sui generis intellectual property protection and the legal requirements for protection under the new legislation.

BACKGROUND

The U.S. Constitution grants Congress the power to enact legislation to promote the progress of science and the useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries. [2] Under this power, Congress has enacted patent legislation to grant inventors exclusive rights to their inventions when they meet the statutory requirements, and has enacted copyright legislation to give authors exclusive rights to their original works of authorship.

Although this Constitutional clause was enacted without debate and there is little history as to the intent of the framers, that intent is quite clear. The purpose of patent and copyright legislation is to foster innovation and creativity through granting authors and inventors a limited monopoly for finite time periods. If this is the intent of the framers, and if the legislation is to have economic justification, the scope of protection for authors and inventors must strike a balance. One the one hand, the limited monopoly must be of wide enough scope and duration

to give inventors and authors the incentive to commit time and resources to creation. On the other hand, the scope and duration cannot be so overly wide as to hinder the dissemination of the works to society. In the realm of patents, an inventor is granted the exclusive right to make, use or sell her invention for a period of seventeen years from the date of the grant of the patent. In return, the inventor must disclose her invention with enough specificity to allow one skilled in the art to practice the invention upon the expiration of the patent. Similarly, under copyright law, an author receives exclusive rights to her original works of authorship for at least fifty years. In return, those works of authorship are dedicated to the public domain upon expiration of the copyright term. As compared to a patent, the much longer term of a copyright is justified by giving the author exclusive rights only to the expression of her ideas, but not the actual ideas.

Computer software is unique because it is cross between a piece of literature and a machine, it is part prose and part invention, it can be aesthetically pleasing yet functional. It is also a multi-billion dollar industry in the United States, and a major United States export.

Most people would agree that some degree of intellectual property protection for computer software is essential to insure that software developers will have a reasonable opportunity to realize a return on their investment in developing software. The proper scope of the

protection is more controversial. Software is different from other inventions because of the ease which it can be copied, and almost everyone would agree that there should be protection against direct literal copying of the code of a program, i.e. pirating. Without protection for literal copying, software developers will not have a reasonable opportunity to realize a return because the cost of copying is very inexpensive in comparison to the cost of development.

More and more in recent years, a large portion of the value of a computer program resides in the non-literal aspects of the program, and not its literal components. For example, someone could copy the look and feel of a popular program, i.e. perform the same function, interact with the user in the same way, and respond to the same user commands, and yet write the program from entirely different source code. Most people would agree that the intellectual property laws should prohibit such look and feel copying, and indeed the courts have generally protected against this type of copying.

On the other hand, intellectual property protection for software must not be overly broad. The protection granted must not hinder competitors from access to programming techniques that are essential to compete for a disproportional length of time. In addition, the term of protection must no be so long that society is prevented from receiving its side of the bargain (the dissemination

of the work without restraint) until after the work has little or no value. Because the state of the art in computer programming is advancing so quickly and works become obsolete within a few years, this consideration requires special attention.

THE PRESENT SCHEME OF INTELLECTUAL PROPERTY PROTECTION

In order to understand the problems associated with the current scheme of protection for software, it is necessary to understand the parameters of the current scheme whereby software developers utilize a overlapping mosaic of copyright, patent, contract and trade secret law to protect their investments.

Copyrights

Over the past twenty years, copyrights have played the most important role in intellectual property protection for software. In 1974, Congress created a special commission to study the issue of intellectual property protection for computer programs. [3] The National Commission on New Technological Uses of Copyrighted Works (CONTU) recommended that copyright protection should be extended to computer programs, and that protection should extend beyond the literal source code of the program. [4] The scope of the protection for non-literal aspects would be determined judicially through emerging case law. [5]

Copyright protection for the source and object code of

a computer program - the literal aspects of the program - is not controversial. Most commentators would agree that the copyright laws should prevent piracy of the actual code of a program and the courts have appropriately found infringement in this type of case.

There is greater controversy surrounding the protection of non-literal aspects of computer programs and the extent of copyright protection for non-literal aspects of computer programs varies has depended upon the court making the determination.

Important cases starting with the Third Circuit's landmark decision in Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc. [6] have reached sometimes inconsistent results attempting to define the proper scope of protection for non-literal aspects of programs. In Whelan, the defendant's program for managing dental lab business functions used some of the same data and file structures as the plaintiff's program. The defendant claimed that there was no infringement because the copyright laws protected only the actual code of the program, not the non-literal data and file structures. The Third Circuit disagreed and stated that copyright protection was available for the structure, sequence and organization of the program, not just the literal components. [7]

Since Whelan, other Circuits have reached inconsistent results when considering cases of alleged copying of non-

literal aspects of programs. Recently, the Second Circuit has addressed this issue in Computer Associates v. Altai, and expressly rejected the broad holding of the Third Circuit in Whelan. [8] The court in Computer Associates adopted a narrower test for assessing substantial similarity in cases of alleged infringement of non-literal aspects of computer programs. As stated by the Second Circuit, a court must apply an "abstraction - filtration - comparison" test to judge substantial similarity. [9] In the first stage, the abstraction stage, the court would break the allegedly infringed program down into its constituent structural parts. [10] Next, the court must filter out from the constituent structural parts those non-protectable parts that are merely incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain. [11] What remains is a kernel of protectable creative expression which is compared to the structure of the alleged infringing program. [12]

Copyright Protection for User Interface Aspects of Programs

As defined broadly, a user interface is what a user must learn in order to operate the computer - it is the language one uses to communicate with the machine. User interfaces include computer languages, the languages one types into a keyboard that are converted into machine readable source code. Common examples are Fortran, Basic, Pascal, C and others. Graphic user interfaces (GUI's),

FRANKLIN PIERCE
LAW CENTER LIBRARY
CONCORD. N. H.

another type of user interface, are icons or menus that are utilized to select items or to otherwise communicate with a computer. Examples of this include the Windows operating system by Microsoft and the Macintosh interface by Apple.

The state of the law regarding user interfaces is uncertain and several courts have reached the undesirable result of granting broad copyright protection to user interfaces. In Lotus Development Corporation v. Paperback Software International, [13] the plaintiff Lotus alleged that the defendant infringed the copyright on the user interface portions of Lotus's popular 1-2-3 spreadsheet when the defendant marketed a spreadsheet with an identical user interface portion. The District Court for the District of Massachusetts held that there was a copyright infringement and set forth a three part test for infringement of user interfaces. [14] First, the court must choose from the alternatives along the scale from the most generalized conception to the most particularized, and choose some formulation - some conception or definition of the idea - for the purpose of distinguishing between the idea and its expression. [15] Second, the court must focus upon whether an alleged expression of the idea is limited to elements essential to expression or instead includes identifiable elements of expression not essential to every expression of that idea. [16] Third, the having identified elements of expression not essential to every expression of the idea, the court must focus on whether those elements are a

substantial part of the allegedly copyrightable work. [17] Applying this test, the court found that the idea of the Lotus 1-2-3 program was to create a spreadsheet to manipulate columns of numbers, and the user interface was a part of the expression of the idea, and found infringement. [18]

In a later case, Lotus sued Borland International, Inc. for infringement of the copyright on its user interface when Borland allegedly included a macro translation feature in Borland's product. [19] By this time, Lotus's spreadsheet product has become a standard in the industry, and the user commands have become very familiar to its users. Borland, a much smaller company attempting to gain market share in the spreadsheet industry, introduced its product which incorporated similarities to Lotus's keystroke sequence and the ability to customize the interface to emulate Lotus's spreadsheet. Applying the test previously set forth, The United States District Court for the District of Massachusetts, per Judge Keeton, held that copyright protection extended to the program's menu structure and organization, and copying by the defendant of the first letters of command names constituted infringement. [20]

In a recent case concerning user interfaces, Apple Corporation sued Microsoft Corporation alleging that Microsoft's introduction of its Windows graphic user interface infringed Apple's copyright on its Macintosh

interface system. [21] This case was tried in the United States District Court for the Northern District of California and the result here seems to be inconsistent with the result reached by the Massachusetts Court in Lotus. The court here held that there was no infringement because there was no virtual identity between the two works. [22] However, the court appears to accept the proposition that the user interface is a copyrightable aspect of a program and left open the possibility of infringement of a user interface.

Patent Protection for Software

Until the 1980's, patent protection for computer software inventions was not available. The Patent Office had a policy of rejecting patent applications which were directed toward software inventions on the theory that programs were merely algorithms and thus non-protectable mathematical formulas.

A new era of software patents began in 1981 with the Supreme Court holding in Diamond v. Diehr. [23] This case involved a process for curing rubber which included one element comprising a computer program. The Court held that process was statutory subject matter for patent protection notwithstanding the presence of the computer program element. [24] Only computer programs which were mathematical algorithms in the abstract were thereafter considered non-statutory subject matter.

It now appears that even inventions that comprise purely computer programs (i.e. algorithms) are patentable if they meet the other requirement of the patent statute. One example is Arrhythmia Research Technology, Inc. v. Corazonix Corporation where the Court of Appeals for the Federal Circuit (the court that now hears all patent appeals) held that a patent directed to computer analysis of electrocardiographic signals of heart functions was statutory subject matter for a patent. [25]

Over the past few years, hundreds if not thousands of patents have been issued with claims directed to subject matter comprising computer programs. There has been a great deal of criticism of the Patent Office for issuing patents where prior art should have prevented the issuance of these patents - the most recent examples are the Stac patent and the Comptons patent described later in this paper.

CONGRESS SHOULD NOW CONSIDER A SUI GENERIS TYPE OF INTELLECTUAL PROPERTY PROTECTION FOR SOFTWARE

Broad Protection for User Interfaces is not Desirable

The arguments against providing broad intellectual property protection for user interfaces, perhaps stated most eloquently by Richard Stallman and the League for Programming Freedom, are very convincing. [26] The most compelling argument is that diversity in user interfaces is not desirable. Users of computer software do not value diversity because of the time that they must invest in

learning a new interface. Therefore, instead of stimulating competition, protection for interfaces discourages it. If an interface is protected by a copyright, a competitor will be forced to develop an entirely different interface in order to introduce a new software product. This will require a user to retrain using the new interface and since retraining requires an investment of time, new users will be unlikely to purchase the competitors product. Users may choose to continue to use a program containing an interface that they are familiar with rather than investing the time to learn a new and perhaps superior product if it contains an unfamiliar user interface.

Software developers do not need the additional incentive to develop user interfaces that is provided by copyright protection for user interfaces. As the League for Programming Freedom points out, the plaintiffs in recent interface copyright law suits (Lotus and Apple) developed their interfaces before copyright protection was provided for user interfaces. Even though competitors were free to copy the interfaces, these companies were very successful and received large returns on their investments.

The test set forth by the Lotus court for infringement of non-literal user interface portions of a program has been criticized on the ground that it takes too narrow a view of what is the idea of a program. [27] The Lotus case itself provides a perfect example. If the idea of the Lotus 1-2-3 program is to provide the user with a spreadsheet to

programs is less appropriate. Today, much of the novelty in programming is embodied in the structure and sequence of new programs, and new programs with novel structure and sequence often borrow code from existing programs. With the evolution of new programming techniques, the comparison of computer programs to works of literary merit has become less appropriate.

Copyright Protection for Software Threatens to Undermine the Tenet that Utilitarian Aspects of Works Are Not Protectable

Since the Supreme Court holding in Baker v. Selden [28] more than one hundred years ago, it has been a basic premise of the copyright laws that expression not ideas are protectable under copyright law, and utilitarian aspects of works are excluded from protection. Copyright protection for software may tend to dilute this basic premise because software is by its nature a utilitarian work. The exception to the exclusion of utilitarian aspects of works from copyright protection may set a regrettable precedent.

Inconsistent Judicial Outcomes Lead to Uncertain Results

This paper has already addressed the inconsistent results that Federal Circuits have reached when required to apply the copyright statute to computer programs. These inconsistent results have led to uncertainty with regard to the scope of protection offered to computer programs,

particularly in the area of protection for non-literal aspects of computer programs. The outcome of a law suit alleging infringement of non-literal structure, sequence and organization aspects of a program may differ depending on the circuit that hears the suit. The Third Circuit applying the Whelan v. Jaslow structure, sequence and organization standard may find infringement where the Second Circuit applying the abstraction, filtration and comparison test may find no infringement. Inconsistent results between the circuits is undesirable because this may lead to forum shopping by plaintiffs.

The Patent Office has had Difficulty Determining Novelty and Non-Obviousness in Software Applications

The Patent Office has consistently had difficulty determining novelty and non-obviousness in software inventions, and some have criticized the Patent Office for issuing absurd patents. There have been cases of patents issued on software inventions where the prior art was not discovered because it was too obvious to publish a paper describing it. The League for Programming Freedom cites the example of a patent issued to AT&T claiming the use of backing store in a window system that lets multiple programs have windows. [29] This same technique had previously been developed at MIT and considered too obvious to publish (so claims the writer for the League for Programming Freedom - himself affiliated with MIT.)

Two recent events further illustrate the problems that the Patent Office has had in examining software inventions. In August of 1993, the Patent Office issued a patent to Comptons New Media that contains claims directed to a search method using a multimedia database consisting of text, picture, audio and animated data. [30] This patent appeared to give Comptons the right to exclude others from using some of the most popular basic searching techniques. When Comptons announced that this patent had been issued, and that it intended to enforce its patent rights, the software industry reacted with outrage. Many people criticized the Patent Office for issuing the patent on the grounds that the techniques were well known in the industry and the invention did not meet the statutory requirements for novelty and non-obviousness. In December of 1993, Bruce Lehman, the Patent Commissioner, took the unusual step of announcing that the Patent Office would reexamine Comptons patent on the basis of newly discovered prior art, and on March 28, 1994 the Patent Office announced the rejection of all claims in the Comptons patent. [31] Under the Patent Office's reexamination procedure, Comptons now has two months to respond to the rejection of the claims. [32]

On February 23, 1994, a federal court jury in San Francisco assessed a record $120 million judgment against software giant Microsoft for infringing a patent owned by a small California company named Stac Electronics. [33] This patent for a data compression apparatus and method contains

claims for converting an input data character stream into a variable length encoded data stream in a data compression system. Software industry experts have questioned the validity of the claims of this patent on novelty grounds as well. Because of the size of the jury verdict, the potential impact on the software industry and the power of Microsoft, this verdict is likely to be appealed.

This problem with patents for software inventions - the inability of the Patent Office to recognize that an applicant's program is anticipated by prior art and therefore granting an absurd patent - is a problem that is might be addressed without making sweeping changes, and in fact the Patent Office has recently taken action to hire more qualified computer scientists. In January of 1994, the Patent Office held hearings in San Jose to discuss the issue of patent protection for software. At these hearings, Bruce Lehman announced the formation of an Electronic Information Center in Group 2300 to improve the collection of software in the Patent Office and to improve access to prior art programs.

In addition, the patent laws already provide a remedy for patents that issued despite prior art which was not discovered. Section 302 of the patent statute allow <u>any</u> person to request the reexamination of an issued patent on the basis of a patent or printed publications. [34]

Nevertheless, because statutory intervention is necessary in light of undesirably broad copyright

protection for user interfaces, it is recommended that intellectual property protection for software be provided with a single unified sui generis format.

SPECIFIC PROPOSAL FOR SUI GENERIS LEGISLATION [35]

This paper advocates new Congressional legislation for sui generis intellectual property protection for computer software. Derived partially from the best aspects of the existing format under copyright and patent law, and partially brand new law, the sui generis legislation would attempt to address the problems with the existing scheme.

A New Federal Agency

This proposal for sui generis intellectual property protection advocates the creation of a new federal agency for the administration of the new legislation. For the purposes of this paper, I shall hereinafter refer to this new agency as the 'Software Office'. The Software Office could be totally independent, or alternatively could be a separate entity of the patent and trademark office. Like other federal agencies, the Software Office would be empowered to issue regulations as permitted by the Congressional enabling statute, subject to judicial review.

The Software Office would be required to hire qualified computer experts in the various fields of computer science (analogous to examining units in the patent office) as the advancing state of computer science

requires. These examiners would carry out prior art searches and to identify novel and non-obvious aspects of applicants' software inventions. In addition, the examiners would be trained in the sui generis software law.

The Software Office would assemble a data base of the existing art all field of computer science. As the state of the art advances, the Software Office would update the data base in order to keep current. The public could be allowed access to this computerized data base in order to permit a member of the public to determine the state of the art. In this manner, a member of the public could make a preliminary determination as to the novelty of his invention. In addition, a member of the public might be allowed to download programs that are in the public domain, perhaps according to a fee schedule with the funds allocated between the software developer and the Software Office. This would contribute to the advancement of the state of software technology because it would eliminate the need to duplicate effort which has already been expended.

Novelty Determination

Similar to the requirement of novelty and nonobviousness under the present patent system, a applicant for registration of a software program would submit her program to the Software Office for determination of novelty and non-obviousness. Upon application to the Software Office, the application would be assigned to an examiner in

an examining unit skilled in that particular art. A program which performs the same function, in the same manner, and in the same way as an existing program would be refused registration for lack of novelty. The program would be refused registration even if the program was written using entirely different code. In addition, similar to the patent regime, a program would not receive intellectual property protection if the program is an obvious variation of the prior art, i.e. if the differences between the applicant's program and the prior art would be obvious to one skilled in the art.

Term of Protection

The term of intellectual property protection for novel and nonobvious software should reflect the social bargain which provides the justification for providing protection. The term must be long enough to provide the programmer with an incentive to write a new program and give the programmer a reasonable opportunity to regain a return on his investment. On the other hand, the term of protection must not be so long as to deny the public access to the program until after the software has become obsolete. Under the current scheme, software protected under patent law is protected for a period of seventeen years from the date that the patent issued. Software protected under copyright law is granted a term of at least fifty years, and in the more usual case of works made for hire - seventy five

years. Because of rapid state of advance of technology in the computer software industry, these terms are far too long since the software becomes obsolete before the expiration of the term of protection.

This paper advocates a much shorter term of protection than is currently provided. A term no longer than five years should give a programmer a reasonable opportunity to receive a return on his investment thereby providing the requisite incentive to create. A truly novel program should retain value after five years and when dedicated to the public domain, the public receives the benefit of its side of the social bargain.

User Interface Aspects of Programs Not Protected Broadly

Under the sui generis legislation for protection of computer software, user interface aspects of computer programs would not be given broad protection.

The first stage in the statutory scheme would be to define the parameters of user interfaces. This would be a matter of negotiation and compromise first in Congressional committee and then in the full House and Senate. As a guideline, this paper recommends the broad definition already delineated - the commands a user must learn in order to communicate with the computer.

The next stage would be to specifically define the scope of protection granted to user interfaces. The scope of protection would also be a matter of Congressional

negotiation and compromise. This paper advocates the complete exclusion of user interfaces from protection. Another alternative would be to grant limited protection to user interfaces, but allow other programmers access to interfaces without fear of infringement upon payment of compulsory licensing fees.

Short of excluding interfaces entirely, or legislating a compulsory licensing arrangement, this sui generis plan for software protection reduces the stifling effect of user interface exclusivity by shortening the term of protection. If an interface becomes an industry standard, and a large number of users are trained to use that particular interface, after the relatively short term of protection the interface becomes part of the public domain.

CONCLUSION

Because of undesirable holdings in several courts that grant broad copyright protection to user interface aspects of computer programs, advancements in programming techniques which render copyright protection for software unsuitable, and the inability of the Patent Office to deal with inventions comprising computer software, it is time for Congress to seriously consider a new sui generis type of intellectual property protection for computer software. This paper advocates the formation of a new federal agency which would oversee the new form of protection. The agency would compile a data base of prior art computer programs,

examine applications for novelty and non-obviousness, and grant protection to those programs that meet the statutory requirements. The term of protection would be short in comparison to the terms issued under the existing copyright and patent formats. User interface aspects of computer programs would be granted either very narrow protection or no protection whatsoever.

Through this new sui generis legislation, a bargain is struck whereby software developers will have the incentive to commit time and resources to advance the state of technology while society obtains access to the software before it becomes obsolete. Through sui generis legislation, the United States can retain its dominant position in software development.

Footnotes

1. Lotus Development Corporation v. Paperback Software International, 740 F.Supp. 37, (D. Mass. 1990)

2. U.S. Constitution, Art. 1, Section 8, clause 8

3. NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS (CONTU) FINAL REPORT 20-21 (1978)

4. Id.

5. Id.

6. Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d. 1222, (3d Cir. 1986)

7. Id. at 1248

8. Computer Associates International, Inc. v. Altai, Inc., 982 F.2d. 693 (2d Cir. 1992)

9. Id.

10. Id.

11. Id.

12 Id.

13. Lotus Development Inc. v. Paperback Software International, 740 F.Supp. 37, (D. Mass. 1990)

14. Id.

15. Id. at 60

16. Id.

17. Id.

18. Id. at 68

19. Lotus Development Corporation, v. Borland International, Inc., 831 F.Supp. 202 (D. Mass 1993)

20. Id.

21. Apple Computer, Inc. v. Microsoft Corporation, 799 F.Supp. 1006 (N.D. Calif. 1992)

22. Id.

23. Diamond v. Diehr, 450 U.S. 175, 101 S.CT. 1048, (1981)

24. Id.

25. Arrhythmia Research Technology, Inc. v. Corazonix Corporation, 958 F.2d. 1053, 22 U.S.P.Q. 1033, (C.A.F.C. 1992)

26. The League for Programming Freedom, Against User Interface, (1991) (Private Paper)

27. See, e.g. Samuelson, Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of Lotus v. Paperback, 55-SPG LCPR 311

28. Baker v. Selden, 101 U.S. 99, 102-106, (1879)

29. The League for Programming Freedom, Against Software Patents, at p. 9 (1991) (Private Paper)

30. Patent Number 5,241,671

31. National Law Journal, January 24, 1994, S.1 (Col. 2)

32. 37 C.F.R. 1.530 (b)

33. National Law Journal, March 7, 1994, P.6 (Col. 1)

34. 35 U.S.C. 302

35. Many of the ideas for this section are taken from: Note, Sui Generis Intellectual Property Protection for Computer Software, 60 Geo. Wash. Law Rev. 997, (1992) (authored by John C. Phillips)