Brooklyn Law Review

Volume 67
Issue 2
THE SECOND CIRCUIT REVIEW: 2000-2001 TERM

Article 10

12-1-2001

The Future of Computing Software in the Reverse Engineering War: Excessive Protection v. Innovation

Barbara J. Vining

THE FUTURE OF COMPUTER SOFTWARE IN THE REVERSE ENGINEERING WAR. EXCESSIVE PROTECTION V INNOVATION*

INTRODUCTION

Computer programs have become increasingly prolific throughout the duration of the ongoing technology boom. With this proliferation, copyright law has been pushed to the limits in an attempt to balance the fear of excessive protection with the desire to encourage innovation. Recently, the scope of these limits has been further challenged by the process of reverse engineering.[1] Reverse engineering of computer software may be accomplished through black box analysis, disassembly, or the clean room process.[2] Of these three methods, disassembly has been at the heart of most recent software litigation. Disassembly involves making intermediate copies of a computer program and subsequently translating the program from machine-readable object code to human-readable source code. Since copyright law gives the copyright owner the exclusive rights to reproduction, derivative works, and distribution,[3] the overarching question is whether the practice of reverse engineering can be accommodated by current copyright laws.[4]

---

[1] The United States Supreme Court has defined reverse engineering as a means of "starting with the known product and working backward to divine the process which aided in its development or manufacture." *See* Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974); *infra* Part I.B.

[2] *See* Betsy E. Bayha, *Reverse Engineering of Computer Software in the United States, the European Union, and Japan*, C137 ALI-ABA 175 (1995).

[3] *See* 17 U.S.C. § 106 (2001).

[4] Reverse engineering does not really raise a question of legality under current copyright law. Legality only becomes an issue to the extent that the ambiguity inherent in the Act makes it unclear whether reverse engineering is legal. The real

This Note analyzes the copyright law issues raised by reverse engineering of computer programs.[5] Section I presents an overview of computer programs and reverse engineering processes and the key legal issues raised. Section II provides an examination of current copyright law governing computer programs, both within the United States and abroad, and how it has developed through relevant case law Section III analyzes the ambiguity surrounding the current body of law and how this ambiguity affects the arguments underlying both the innovation and protection premises. Finally, Section IV of this Note posits an amendment to current copyright law by setting limits on the reverse engineering process. In particular, this Note proposes that current copyright laws be amended to address reverse engineering by expressly advocating black box analysis and allowing disassembly only as a function of the clean room process. The proposed statutory guidelines are tailored to permit enough reverse engineering to encourage and allow innovation of computer software while providing copyright owners with a modicum of additional protection. By explicitly allowing reverse engineering in a modified form, the proposal effectively encourages both innovation and competition without bestowing copyright owners with excessive protection, thus allowing the software industry and the public to benefit.

---

question is whether the Act was intended by Congress to include and allow reverse engineering or whether the Act's silence on the matter reflects no conferral of a reverse engineering right in respect to computer software.

[5] Although this Note addresses reverse engineering only with respect to its effect on competition within the computer software market, reverse engineering does serve other functions. Reverse engineering is used, for example, to teach students how to write programs, to repair malfunctioning software, and to modify a program for one's own computer. *See, e.g.*, Brian C. Behrens & Reuven R. Levary, *Legal Aspects—Software Reverse Engineering and Copyright: Past, Present and Future*, 31 J. MARSHALL L. REV. 1 (1997).

I. BACKGROUND

A. *Computer Programs*

Computer programs are created by writing line-by-line instructions directing the operation of the computer. Whether the instructions are written alphanumerically or strictly numerically, a computer programmer is ultimately able to instruct a computer, through writing and combining lines of instructions, to perform various tasks and execute various functions.[6]

When displayed alphanumerically, these instructions are collectively referred to as "source code."[7] Source code is written in a high-level computer programming language that can be read and understood by humans.[8] In addition to the actual instructions that comprise the final program, a programmer writing in a computer programming language can add comments and labels to procedures within the program. The advantage of using comments and labels is that the functionality of specific instructions, or the thoughts of the programmer, can be documented for future reference or modification.

Upon completion, the source code is subsequently compiled into "object code" that is understood and executed by a computer.[9] Object code is displayed simply as binary "zero" and "one" digits. The purpose of this binary code is to relay to a computer whether switches in the computer chip should be

---

[6] For example, a line of code might instruct the computer to multiply any two digits entered from the keyboard and to display the results. Or, a simple instruction might be one that tracks the movement of the mouse pointer and displays its coordinates on the screen for the user to follow. A computer program integrates instructions and allows the computer to perform larger functions, such as word processing, or may even create an entire operating system.

[7] For a thorough discussion of software development, see Andrew Johnson-Laird, *Reverse Engineering of Software: Separating Legal Mythology from Actual Technology*, 5 SOFTWARE L.J. 331, 336 (1992).

[8] Some of the more common high-level computer programming languages include COBOL, FORTRAN, BASIC and C. *See id.*

[9] *Id.* Based on the programming language used, a corresponding program will compile the alphanumerical source code written by the programmer into the binary object code used by the computer.

"off" or "on," respectively.[10] Each individual line of alphanumeric source code is converted into corresponding numerical strings of zeroes and ones upon compilation. A compiled program appears as a continuous series of binary digits covering numerous (usually, hundreds or thousands of) pages, and is thus overwhelming and often incomprehensible for humans. Thus, while a programmer could write an entire program in binary code, programmers normally write computer programs in a high-level computer programming language.

B. *Reverse Engineering Processes*

Since the actual source code of a functioning program is not often available, programmers frequently engage in reverse engineering to gain knowledge about how a particular program functions. A programmer can utilize reverse engineering to create an interoperable or competing program by determining the specifications and functions of the original program. Whether the ultimate goal is to create a new program to interoperate with existing programs,[11] or to compete for the same users by producing either an improved or altogether new program, numerous methods of reverse engineering exist.[12] While some methods of reverse engineering are more legally

---

[10] *See* Greg Weiner, *Reverse Engineering as a Method of Achieving Compatibility in the Computer Industry*, 6 U. BALT. INTELL. PROP. L.J. 1, 2 (1997).

[11] For example, a programmer may reverse engineer the program that operates a video game console in order to create a new video game that can run on the existing console. By determining how the particular console functions, the programmer can write new code using the specifications necessary to allow the game and console to interoperate. In order for any two programs to achieve interoperability, the proper interfaces and protocols must be accurately used. *See* Johnson-Laird, *supra* note 7, at 338.

[12] Two classic techniques are black box reverse engineering and the process of dismantling, testing, and observation. Additionally, there are various ways to read programs stored in computers or on other media out of the machine into a form that is meaningful and readable by a human being, including reading specifications and manuals, reading stored programs with a debugger program, reading ROM's, disassembly, and the clean room process. *See* G. Gervaise Davis, III, *Scope of Protection of Computer-Based Works: Reverse Engineering, Clean Rooms and Decompilation*, 370 PRACTISING LAW INSTITUTE: PATENTS, COPYRIGHTS, TRADEMARKS, AND LITERARY PROPERTY COURSE HANDBOOK SERIES ("PLI/PAT") 115, 145 (1993). This Note will only address the more relevant and abundant reverse engineering methods of computer software: black box analysis, disassembly, and clean room process.

controversial than others, there are key legal issues common to all reverse engineering processes.

1. Black Box Analysis

Black box analysis is a classic method of reverse engineering that allows programmers to discover the functioning of a program through extensive observation. The desired information about how a device functions is obtained by inputting data to the device and then observing the outcome.[13] Through the iterative process of altering the input and observing the result, it may be possible to ascertain how the original device functions. In this way, later users can determine their own way of accomplishing the same functions without ever having looked at the underlying source or object code of the software involved. [14] Moreover, the reverse engineer avoids the use of intermediate copies of the software of the original device.[15] Although not challenged by allegations of illegality, black box analysis is so simplistic as to be rather impractical. In light of the complexity and length of today's computer programs, it is oftentimes not feasible or possible to determine the functioning of a program by mere observation.

2. Disassembly

Disassembly is an increasingly common and effective way to reverse engineer computer software. Ultimately represented as binary or hexadecimal digits in ROM or disk version, programmers engage in disassembly to gain access to the source code in which the original program was written.[16] By translating backwards from virtually unintelligible object code to more easily understood source code, the reverse engineer can read and study the original program. The process

---

[13] *See* Bayha, *supra* note 2, at 179.

[14] *See id.*

[15] *See id.*

[16] Most computer programs are stored on disk or in ROM in the hardware. The ROM or disk version is simply a physical representation of binary digits that must be converted into a human readable form for study. Davis, *supra* note 12, at 148.

of disassembly involves copying the object code version of a program from the ROM or disk version onto the screen or onto paper. The object code is then converted, or disassembled, into a sort of "virtual source code."[17] Although this virtual source code is more easily read by humans than the object code, it is vastly different from the original source code written by the software's designer.[18] When the source code is initially compiled, only those lines which contain instructions to the computer are converted to object code; any of the original comments and labels that the programmer may have used to explain certain aspects of the program or to identify parts of the program are lost and thus unavailable upon disassembly [19] In comparison to black box analysis, with an increase in practicality and effectiveness comes an increase in challenges of illegality Unlike black box analysis, disassembly inexorably involves the making of interim copies of copyrighted computer programs.

3. The Clean Room Process

The clean room process is a method of reverse engineering that may employ black box analysis or disassembly or both.[20] The first step is to develop a set of high level specifications or criteria necessary to develop a program compatible with or functionally identical to the original program.[21] This information is gleaned from available manuals and specifications or possibly by performing black box analysis. Additionally, it may be necessary to disassemble the original program in order to facilitate the analysis. The resulting

[17] Bayha, *supra* note 2, at 180.

[18] "The metamorphosis of human readable source code into computer executable programs strips off too much information for it to be reversed." Johnson-Laird, *supra* note 7, at 346.

[19] Programmers often include comments in the source code during the development of a program to explain what certain portions of code do or to signal potential problems in the code for later reference. Labels are often chosen and used by programmers to set-off portions of the code or as obvious "dummy" names for different functions of the program. Dummy names are labels given to specific procedures or functions that denote its obvious purpose, i.e., a procedure named AddAllEvenNumbers, which adds all even numbers entered by the user.

[20] *See* Bayha, *supra* note 2, at 180.

[21] *See id.*

specifications or criteria are to be free from any of the "expression" of the original program or decompiled code.[22]

The first step is achieved by reverse engineer programmers who are considered to be "dirty," having studied the original program and written this set of guidelines.[23] The next step is to hand off these specifications to a team of "clean" programmers who have never seen or been exposed to the original program.[24] This clean team now programs new code from the specifications and criteria based only upon the high level specifications. The "clean" and "dirty" programmers must not be permitted to communicate directly.[25] By prohibiting the two independent teams of programmers from direct interaction, the clean programmers presumably will not be tainted by access to the original program. If and when questions arise, they are dealt with in writing and through an intermediary, usually a lawyer knowledgeable of reverse engineering and copyright law issues.[26] Considering that this reverse engineering process can, and often does, involve disassembly, it too faces the consequent legal issues.

4. Legal Issues Inherent in All Reverse Engineering Processes

The basis of most legal questions surrounding reverse engineering arises from the fact that the process requires reading, which requires copying. Reading an existing, and presumably copyrighted, computer program requires making a copy of it, first in the memory of the computer and then on the screen or on paper.[27] Reverse engineering also raises legal issues regarding the right to make emulations and interim copies.[28] At minimum the right to read and to copy, if necessary to read, appears to be a right recognized by both the courts and

---

[22] *See id.*
[23] *See* Davis, *supra* note 12, at 151.
[24] *See* Bayha, *supra* note 2, at 181.
[25] Davis, *supra* note 12, at 151.
[26] *See id.*
[27] *See supra* Part I.B.2.
[28] *See* Davis, *supra* note 12, at 157.

the legislature.[29] In the subsequent sections of this Note, arguments for both the legality and illegality of reverse engineering of computer software will be addressed and analyzed.

II. COPYRIGHT LAW AND COMPUTER PROGRAMS

A. *Statutory Developments*

1. The United States

While there are numerous ways in which programmers can protect computer software, the main vehicle for preventing the copying of a computer program is through the protections provided by copyright law.[30] Computer programs have been copyrightable subject matter since 1978, when the Copyright Act of 1976 became fully effective.[31] Congress further amended the Copyright Act in 1980 to explicitly reconfirm that computer programs are copyrightable subject matter.[32] Under the

---

[29] *See id.* at 154-55 (citing Justice O'Connor and the authors of the CONTU Report (for a description of CONTU, see *infra* note 32)).

[30] Other means of protection for computer programs are outside the scope of this Note. *See, e.g.*, Ronald L. Johnston & Allen R. Grogan, *Trade Secret Protection for Mass Distributed Software*, 11 COMPUTER LAW. 1 (1994) (discussing the case for trade secret protection for the internal design of widely distributed software); Mark A. Lemley & David W O'Brien, *Encouraging Software Reuse*, 49 STAN. L. REV. 255 (1997) ("the primary means of legal protection for computer software has shifted from copyright to patent").

[31] *See* H.R. REP. No. 1476, at 116 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5731, 5731 (stating that literary works protected under section 102(a)(1) of the Copyright Act include computer programs) [hereinafter H.R. 94-1476].

[32] Congress created a committee, the National Commission on New Technological Uses of Copyrighted Works ("CONTU"), to analyze the law in light of emerging technologies and to propose amendments to the copyright law that would better protect computer programs. In accordance with CONTU's recommendation that computer programs be added to § 101 of the Copyright Act, Congress amended the Copyright Act in 1980 to explicitly reconfirm that copyright protection extends to computer programs by adding the definition of "computer programs" to § 101. In addition, CONTU recommended the substitution of a new § 117 that now awards copyright owners of computer programs a right to make limited adaptations or modifications to their program and to make back-up copies of the program. *See*

current statutory scheme, the Copyright Act provides protection for "original works of authorship fixed in any tangible medium of expression."[33] These "original works" include "literary works," which are defined as "works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects in which they are embodied."[34] Although computer programs are not explicitly included in the statutory definition, it is well established that computer programs are protected as literary works.[35]

The Copyright Act provides protection only for the expression of ideas, not for the ideas themselves.[36] This concept has been codified in § 102(b) of the Copyright Act, which provides that "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such a work."[37] Therefore, subsequent users are free to copy and utilize the ideas in a copyrighted work, provided the original expression is not likewise appropriated.[38]

---

generally CONTU Final Report on New Technological Uses of Copyrighted Works (1979) [hereinafter CONTU Final Report] (on file with author); Pamela Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471, 474-75 n.12 (1985).

[33] 17 U.S.C. § 102(a) (1990).

[34] 17 U.S.C. § 101 (1990).

[35] *See supra* note 31 and accompanying text.

[36] For example, think of the well-known video game Pac-Man. Expression of the particular characters are protectable and would not have been created but for the original programmer. However, the idea of a character navigating around a maze and collecting various items, while avoiding adversaries, is not protectable. This concept is called the idea-expression dichotomy. *See* Baker v. Selden, 101 U.S. 99, 102-03 (1879). Congress has made clear that this dichotomy applies with equal force to computer programs. *See* H.R. REP. No. 94-1476, *supra* note 31, *reprinted in* 1976 U.S.C.C.A.N. at 5670. As a matter of policy, creative expression is protected by copyright to encourage initial innovation while ideas are not protected by copyright to encourage subsequent innovation and competition. *See* discussion *infra* Part III.

[37] 17 U.S.C. § 102(b) (1994).

[38] Exceptions to this general rule are the doctrines of merger and *scenes á faire*. If only one way, or a limited number of ways exist to express an idea, the idea and the expression are considered to have merged and there is thus no protectable expression. *Scenes á faire*, on the other hand, denies copyright protection to literary elements that are considered to be inevitably associated with a theme or plot. *See* Arthur W. Fisher & Yvonne Reyes, *Copyright Protection for Computer Software*, 477

In the case of computer programs, it is especially difficult to bifurcate the idea-expression dichotomy.[39] This fundamental determination initially depends upon the ability to separate the literal and nonliteral elements of a computer program.[40] The literal code of a program is considered protected expression. Verbatim copying of literal code is thus a copyright infringement. Nonliteral, or functional, elements of a computer program, generally referred to as the structure, sequence, and organization ("SSO"),[41] are protected by copyright laws as well. One of the purposes of protecting nonliteral elements of a program is to prevent later developers from evading infringement by simply making minute changes. The overall determination of infringement necessarily relies upon the degree of similarity two programs may have.[42] Similarity is based on the extent to which the intellectual work in one program can be used in a subsequent program. The line drawn between protected and unprotected elements is ultimately a determination of the desired level of competition within the software industry.[43] Articulating this line has proven elusive.

PLI/PAT 439, 445 (1997).

[39] *See generally* Honorable Jon O. Newman, *New Lyrics for an Old Melody: The Idea/Expression Dichotomy in the Computer Age*, 17 CARDOZO ARTS & ENT. L.J. 691 (1999).

[40] For an in-depth analysis of the problems associated with distinguishing literal and nonliteral elements of computer programs, see Bruce G. Joseph, *Copyright Protection of Computer Software and Complications*, 602 PLI/PAT 209 (2000) (providing a thorough review of how courts have sought to draw the line).

[41] Some examples of nonliteral elements include data structures, file structures, sequences for communicating between programs, programs that govern other program modules, and expressive elements of a program's user interface, which in turn includes menus, screen displays, and icons, to name a few. *See* Fisher & Reyes, *supra* note 38, at 447.

[42] *See infra* note 64 and accompanying text.

[43] A line drawn heavily toward either end of the spectrum would harm the industry by creating too great an effect on the level of competition. Conversely, a line drawn carefully closer to the protectionist or non-protectionist end of the spectrum would scarcely diminish or enhance competition, respectively. If, for example, the line were drawn increasingly closer to the non-protectionist end, this would signal to the industry that a high level of competition is desired. *But see* discussion *infra* Part III.C. (explaining how more protection effectively leads to increased competition by encouraging programmers to create new works in the first instance).

2. The European Community's Software Directive

In the late 1980s, the European Community Commission began a project to provide legal protection for computer software in the member states of the European Union ("EU").[44] The project resulted in the issuance of the Software Directive in May 1991.[45] Article 1 of the Directive provides that "[m]ember [s]tates shall protect computer programs, by copyright, as literary works within the meaning of the Berne Convention for the Protection of Literary and Artistic Works."[46] During its consideration of the Software Directive, the Commission grappled with the contentious issues of reverse engineering and interoperability.[47] Despite the intense lobbying that surrounded the Commission's dealings, the Software Directive reflected a compromise.[48]

In addressing the issues of reverse engineering and interoperability, the Software Directive contains a specific provision. Article 6 permits decompilation,[49] through reproduction and translation, as a last resort to achieving interoperability, subject to three conditions: (i) that "these acts are performed by the licensee or by another person having a right to use a copy of the program, or on their behalf by a person authorized to do so";[50] (ii) that "the information

---

[44] Now called the European Union, it was known as the European Community when Software Directive was issued. It will be referred to as the European Union ("EU") throughout this Note, except when referenced in cited material.

[45] *See generally* Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs, 1991 O.J. (L 122) 42 [hereinafter Software Directive].

[46] *Id.* at art. 1.

[47] For an explanation and example of interoperability, *see supra* note 11 and accompanying text.

[48] *See* Vanessa Marsland, *Copyright Protection and Reverse Engineering of Software—An EC/UK Perspective*, 19 U. DAYTON L. REV. 1021, 1031 (1994) (discussing the industry and lobby pressures and the resulting compromise); Bayha, *supra* note 2, at 188. In particular, Bayha points out that "[i]n general, the U.S. interests tended to favor strong intellectual property protection for computer software, including a prohibition on decompilation." *Id.*

[49] There is no such process as "decompilation." The frequently used term resulted from its use throughout the Software Directive, which was originally drafted in French, since there is no other French term for the disassembly process. *See* Davis, *supra* note 12, at 149.

[50] Software Directive, *supra* note 45, at art. 6.

necessary to achieve interoperability has not previously been readily available";[51] and (iii) that "these acts are confined to the parts of the original program which are necessary to achieve interoperability."[52] Furthermore, Article 5 allows a lawful acquirer to reproduce and translate the software for use or error correction without requiring authorization by the rightholder.[53] The Software Directive even contains a provision expressly permitting black box analysis:

> The person having a right to use a copy of a computer program shall be entitled, without the authorization of the rightholder, to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.[54]

The Software Directive addresses reverse engineering rights and limitations more explicitly than does current copyright law in the United States. Some argue that the rights granted to members of the EU are more restrictive of reverse engineering than rights granted by present U.S. case law.[55] However, the catalyst of this argument is the underlying statutory ambiguity.[56] Although numerous questions remain regarding the scope of reverse engineering, the Software Directive has at least taken a step that Congress has not yet taken; it attempted to statutorily define and limit certain aspects of reverse engineering in response to the concerns of the industry and the public.

---

[51] *Id.*

[52] *Id.*

[53] *Id.* at art. 5.

[54] *Id.* at art. 5.3.

[55] *See* Bayha, *supra* note 2, at 189 (citing Thomas Heymann, *The International Effect of the EU Restrictions on Reverse Engineering*, 2 THE INT'L COMPUTER LAW. 15 (July 1994)).

[56] *See* discussion *infra* Part III.

B. *Case Law*

Three significant appellate court decisions, arising from litigation over video games, specifically address reverse engineering in connection with computer software.[57] All three cases originated in the District Court for the Northern District of California and provide guidance to the courts and the industry on the subject of reverse engineering and the applicability of copyright protection. On both sides of the controversy are also several significant district court decisions that present arguments often relied upon by proponents and opponents of reverse engineering.[58] Additionally, two other unrelated cases have proved instructive as to the general relationship between technology and copyright law [59]

1. Technology and Copyright Law

The Supreme Court's decision in *Feist Publications, Inc. v. Rural Telephone Service Co., Inc.*[60] is easily the most significant case on compilation copyrights and has implications that directly affect copyright protection of technology In a unanimous decision, the Court held that "white pages" telephone listings are not protected by copyright.[61] In making this determination, the Court recognized two well-established propositions—facts are not copyrightable, but compilations of facts generally are.[62] Since originality is a constitutional

---

[57] *See infra* notes 93-95 and accompanying text.

[58] *See* Part II.B.2.

[59] *See, e.g.*, Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340 (1991); Computer Assoc. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

[60] 499 U.S. 340 (1991), *rev'g* 916 F.2d 718 (10th Cir. 1990).

[61] In *Feist*, the plaintiff, Rural Telephone Service Co., was a local telephone company that published a white pages directory pursuant to state law. The defendant, Feist, wanted to create a competing area-wide directory. After plaintiff refused to license its listings to Feist, Feist nonetheless copied plaintiff's directory and compiled its own directory based on reorganizations of and necessary changes to plaintiff's listings. *Id.*

[62] *Id.* at 344. *Feist* provided the principle that not all copyrighted works are accorded the same level of protection. Some portions of a copyrighted work may fall entirely outside the scope of copyright protection if they are facts, ideas, functional elements, or unavailable without infringing the original work. *See also supra* text accompanying note 36.

requirement for copyright protection,[63] and facts are never original,[64] the required originality of a compilation can only be claimed in the way the facts are presented. Thus, in order "to merit protection, the facts must be selected, coordinated, or arranged 'in such a way' as to render the work as a whole original."[65]

The implications of the *Feist* decision affect technology in its applicability to computer databases. A computer database is valued for its selection, inclusion and arrangement of otherwise overwhelming amounts of data. Similar to directories of the type at issue in *Feist*, databases are simply compilations of preexisting material not itself subject to copyright protection.[66] The main problem facing computer databases is, likewise, whether the selection is creatively original. Furthermore, computer databases generally arrange the data so as to optimize search and retrieval functions. Such an arrangement faces the problem of being viewed as entirely typical and devoid of even the slightest trace of creativity, as was the alphabetical listing of data in *Feist*.[67]

Along with *Feist* came *Computer Ass'n. Int'l, Inc. v. Altai, Inc.*,[68] in which the Second Circuit devised a three-part test to determine the occurrence of software infringement. The results of applying the abstraction-filtration-comparison test would determine whether substantial similarity of protectable portions of software programs exists to warrant a finding of

---

[63] *Id.* at 346.

[64] *Id.* at 358.

[65] *Feist*, 499 U.S. at 358.

[66] *See id.* at 361.

[67] *See id.* at 362.

[68] 982 F.2d 693 (2d Cir. 1992), *aff'g* 775 F Supp. 544 (E.D.N.Y. 1991). The plaintiff, Computer Associates, developed a core program that could be modified to communicate with different operating systems. Defendant and competitor, Altai, desired to develop a similar program and hired a programmer who had previously been employed by plaintiff. Unknown to Altai, the hired programmer had taken and used copies of plaintiff's source code to write the similar program for Altai. Upon discovery, Altai set up a team of programmers, not including the previous Computer Associates programmer, to write a replacement program that did not infringe the plaintiff's program. Although the replacement was completed, and subsequently distributed, by the team who had no access to either the purloined source code or the infringing code, plaintiff contended that even the replacement was similar and infringed upon its copyright.

copyright infringement.[69] In addition to application of this test, the court agreed to use a technical expert who emphasized the distinction and the necessity to look to the similarities in the code as opposed to what it was the program did.[70] In accepting the expert advice, the court concluded that there was no substantial similarity between the original and alleged infringing code.[71] In addition to the importance of establishing and encouraging the three-part test, the court also showed that it is possible to correct an infringement by rewriting code without reference to the original infringing work, thus eliminating an ongoing infringement.[72]

The *Feist* and *Altai* decisions were significant mainly because the courts recognized that functional elements of computer programs are not considered creative expression protectable by copyright law While not a computer software case, the principles underlying the Supreme Court's decision in *Feist* apply equally to many elements of computer software design and content. In the arena of software, routine programming practices and obvious methods of programming are not protected. The *Altai* decision specifically acknowledges that such practices and methods are outside the scope of copyright protection. *Feist*, together with *Altai*, has a significant impact on the realm of computer software.

2. Reverse Engineering and Computer Software

a. District Courts

In the early stages of the reverse engineering war, some district courts were reluctant to sanction the processes of reverse engineering. For example, in *Hubco Data Products*

---

[69] *See* Weiner, *supra* note 10, at 16 (providing a concise overview of the *Altai* three-part software infringement test).

[70] *See Altai*, 982 F.2d at 712-13.

[71] *See id.*

[72] In other words, although Computer Associates conceded that the original version of its analogous program was infringing, over thirty percent having been copied from Altai's source code, it was able to rewrite the code independent of the infringing code and avoid a finding of copyright infringement.

Corp. v. Management Assistance, Inc.,[73] the court held reverse engineering of computer programs to be illegal under copyright law In *Hubco*, the copyright owner, MAI, had developed several versions of its operating system, each more costly, which were sold at different prices and with different capabilities.[74] The defendant and alleged infringer, Hubco, had been an MAI dealer who discovered that a routine existed within each different operating system that controlled the speed of the system.[75] Hubco then tried to market its own program that would remove the slow down routine in the operating systems with lower capabilities, thus increasing the speed of the system without the need to purchase the more expensive versions of the system.[76] To determine the speed control routines within MAI's operating systems, Hubco ran comparison tests that inevitably involved reverse engineering.[77] The *Hubco* trial court enjoined this activity on the ground that the Hubco software was an unauthorized derivative.[78]

Similarly, the district court in *SAS Institute, Inc. v. S&H Computer Systems, Inc.* determined that reverse engineering violated copyright law [79] In *SAS*, the defendant, S&H, attempted to fraudulently obtain a source code license to the SAS mainframe program in order to create its own version that would run on an IBM mainframe, as opposed to the DEC VAX platform on which the SAS program was designed to run.[80] The court found that S&H had obtained a license under false pretenses. Although the S&H programmers attempted to disguise and edit out as much of the SAS code as possible, more than forty-four portions of the code were exact copies of that originally written by SAS.[81] Both the *Hubco* and *SAS* decisions exemplify courts' initial reactions to the practice of reverse

---

[73] 219 U.S.P.Q. 450 (D. Idaho 1983).
[74] *See id.* at 452.
[75] *See id.*
[76] *See id.*
[77] *See id.*
[78] *See Hubco*, 219 U.S.P.Q. at 456.
[79] 605 F Supp. 816, 830 (M.D. Tenn. 1985).
[80] *See id.* at 820.
[81] *See id.* at 830.

engineering; in the beginning, courts were averse to allowing the practice under copyright law

Despite the seeming reluctance of some courts to embrace the practice of reverse engineering, other courts, while not explicitly sanctioning it, recognized the possibility that reverse engineering might not always be unauthorized. For example, in *E.F Johnson Co. v. Uniden Corp. of America*,[82] the district court found that defendant, Uniden, had infringed plaintiff's, EFJ's, copyrighted computer software. EFJ had developed a logic trunked radio system ("LTR") primarily for use in motor vehicles.[83] The software allows the LTR system to make "all assigned radio channels accessible to all system users."[84] Thereafter, Uniden reverse engineered the LTR code so as to create its own version of code compatible with the radios in EFJ's LTR system.[85] Despite Uniden's effort to create independent code, the court found substantial similarity between Uniden's code and EFJ's copyrighted code. However, although the court held defendant liable for infringement, it approved of reverse engineering in dicta.[86] In its recognition of reverse engineering as an important practice in the industry,[87] the *E.F Johnson* decision was at the forefront of the legitimization of disassembly of computer programs.

Jumping on the reverse engineering bandwagon, *NEC Corp. v. Intel Corp.* continued the trend of acceptance.[88] In *NEC*, a software engineer at NEC reverse engineered Intel's 8086 and 8088 microprocessor microcode in an attempt to develop a new microcode for use in NEC's comparable microprocessor.[89] Through a process of disassembling and studying Intel's microcode, the engineer created a final version for NEC's V20 and V30 microprocessors. Intel claimed that

---

[82] 623 F Supp. 1485 (D. Minn. 1985).

[83] *See id.* at 1487.

[84] *See id.*

[85] *See id.* at 1490.

[86] *See id.* at 1501-02 n.17 ("The mere fact that defendant's engineers dumped, flow charted, and analyzed plaintiff's code does not, in and of itself, establish pirating. While defendant may have permissibly dumped, flow charted, and analyzed plaintiff's code, it could not permissibly copy it.").

[87] *Id.* (recognizing that "dumping and analyzing competitors' code is a standard practice in the industry").

[88] 10 U.S.P.Q.2d 1177 (N.D. Cal. 1989).

[89] *See id.*

both programs made use of similar "patches" in their respective microprocessors.[90] Notwithstanding similarities in the programs, the court determined that NEC's code did not infringe on Intel's microcode, stating that "when considered as a whole, [NEC's code] is not substantially similar to the Intel microcode within the meaning of the copyright law"[91] Furthermore, the court reasoned that any similarities were simply the product of both programs being subject to the same constraints.[92] The *E.F Johnson* and *NEC* decisions support the trend towards encouraging reverse engineering of copyrighted computer programs. Following these two cases, the courts' implicit approval of disassembly and intermediate copying, within the bounds of the substantial similarity rule, became increasingly apparent.

b. Circuit Courts

The next battle in the reverse engineering war occurred in the more recent cases of *Atari Games Corp. v. Nintendo of America, Inc.*,[93] *Sega Enterprises Ltd. v. Accolade, Inc.*,[94] and *Sony Computer Entertainment Inc. v. Connectix Corp.*,[95] all applying Ninth Circuit law In each of these cases, the appellate courts analyzed the facts under the fair use doctrine to determine whether the reverse engineering methods employed constituted fair use.[96]

---

[90] A patch is used in creating an interrupt sequence in computer code to overcome a detected "bug" in the code. Presumably, Intel was trying to argue that use of such a specialized portion of code evinces that NEC must have copied a significant portion of the overall code if NEC felt it needed that patch to overcome the same bug that Intel had found. *See id.* at 1185. Intel's argument may be analogous to the practice of planting "seeds" consisting of erroneous or inaccurate information in a writing to trap an infringer. *See* 1 PETER D. ROSENBERG, PAT. L. FUNDAMENTALS § 5.01[1], 5-9 (2d ed. 2000).

[91] *NEC*, 10 U.S.P.Q.2d at 1185.

[92] *See id.* at 1188.

[93] 975 F.2d 832 (Fed. Cir. 1992).

[94] 977 F.2d 1510 (9th Cir. 1992).

[95] 203 F.3d 596 (9th Cir.), *cert. denied*, 531 U.S. 871 (2000).

[96] 17 U.S.C. § 107. Section 107 sets forth four factors in determining whether the use of a work is a fair use: (i) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (ii) the nature of the copyrighted work; (iii) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (iv) the effect of the use upon

To the extent that the Court of Appeals for the Federal Circuit held that reverse engineering and intermediate copying are not an infringement when the purpose is to understand the ideas and unprotected elements in a computer program, the *Atari* case is consistent with the later *Sega* decision and the *Altai* case. The Federal Circuit held that "reverse engineering object code to discern unprotectable ideas in a computer program is a fair use."[97] In *Atari*, the defendant, Atari, "peeled" the ROM that contained Nintendo's copyrighted program and then disassembled the contents of the ROM in an attempt to determine the workings of the Nintendo Entertainment System ("NES").[98] In order to prevent unlicensed production of games for the NES, Nintendo developed a highly sophisticated electronic lock and key system that locks out all non-licensed video game cartridges not containing the electronic key.[99] Despite Atari's reverse engineering attempts, it was unable to decipher the code and resorted to obtaining an unauthorized copy of the source code from the U.S. Copyright Office under false pretenses.[100] The court held Atari infringed Nintendo's copyright protection based on the purloined copy of code. However, the court was clear in stating that, under Ninth Circuit law, "[r]everse engineering, untainted by the purloined copy of the 10NES program and necessary to understand 10NES, is a fair use."[101]

The Ninth Circuit recognized the importance of disassembly in the software industry in both *Sega* and *Sony*.[102] Like the Federal Circuit in *Atari*, the Ninth Circuit justified the practice as fair use when reverse engineering was used in an attempt to create a comparable, competing product. In *Sega*, the court applied the fair use analysis where the defendant,

---

the potential market for or value of the copyrighted work.

[97] *Atari*, 975 F.2d at 843.

[98] *See id.* at 836.

[99] *See id.*

[100] *See id.*

[101] *Id.* at 843.

[102] For a comparative analysis of the fair use doctrine in the *Sega* and *Sony* decisions, see Ivan Rothman, *From Sega to Sony and Beyond: An Alternative Legal Basis for Software Reverse Engineering*, 18 INTELL. PROP. L. NEWSL. 3 (Spring 2000). The dilemma, as set forth by Rothman, is that if intermediate copying is prohibited, the public is denied access to unprotected functional elements. On the other hand, if intermediate copying is allowed, copyright owners' basic rights are violated. *Id.*

Accolade, reverse engineered Sega's video game cartridges in an attempt to produce its own video game that would be compatible with Sega's platform.[103] The court accepted Accolade's fair use defense, holding that "where disassembly is the only way to gain access to the ideas and functional elements embodied in a copyrighted computer program and where there is a legitimate reason for seeking such access, disassembly is a fair use of the copyrighted work, as a matter of law "[104] The court's ruling was, however, conditioned on two factors: disassembly must be the only way to gain access to unprotected ideas and functional elements; and such access must be sought for a "legitimate reason." Following *Sega*, alleged infringers can claim a right to engage in intermediate copying and disassembly in these certain limited circumstances.

Similarly, the Ninth Circuit followed this approach in *Sony* [105] And likewise, the court concluded that the four fair use factors favored the defendant and alleged infringer, Connectix. Notwithstanding any factual dissimilarities,[106] the court went beyond its earlier decision in *Sega*, effectively eradicating the *Sega* rule's "limited circumstances" approach. Despite the "legitimate reason" applied to Accolade's fair use defense, the court ruled in *Sony* that a final product which does not contain any code of the original product is transformative and, as such,

---

[103] Sega developed a video entertainment system, Genesis III, which employed a protective system in response to counterfeiters. The system, designed to protect its trademark rights, was a patented process by which the console's operating system reads a game program for specific computer code. If a game program contains the patented initialization code, the Genesis III console would allow the game to be run. Accolade disassembled Sega's cartridges to obtain this initialization code so that it could develop and market Genesis-compatible video games. *Sega*, 977 F.2d at 1520.

[104] *Id.* at 1527-28.

[105] *Sony*, 203 F.3d at 596. In *Sony*, after having been refused a license, Connectix reverse engineered the read-only memory chip used in Sony's video entertainment console to create its own operating system. By employing black box analysis and disassembly, Connectix was able to copy and use the contents of Sony's read-only memory chip. As a result, Connectix developed a software product that emulates both the hardware and software components of the Sony console and enables users to play Sony games on a computer rather than on a television.

[106] For example, unlike the defendant in *Sega*, Connectix did not develop video games compatible with Sony's platform, but an alternative platform compatible with Sony's games. Also, whereas the defendant in Sega exclusively employed disassembly, Connectix relied primarily on black box reverse engineering.

does not supplant the original product nor cause a substantially adverse impact on the potential market of the original product. This line of reasoning suggests that intermediate copying necessary to access and examine unprotected ideas may be sanctioned only if the final product does not infringe the original product. Given the Ninth Circuit's rulings in *Sega* and, subsequently, in *Sony*, the restrictions on intermediate copying of copyrighted computer programs have practically dwindled to nonexistence. In reconciling the fair use defense, the *Sega* ruling, and the significant departure from both in *Sony*, it seems clear that a new approach to analyzing reverse engineering of computer software is crucial.[107]

From *Hubco* to *Sony*, courts have been faced with the arduous challenge of balancing the need to protect computer programs against the need to encourage competition and promote innovation. With an inadequate statutory framework, the vast majority of cases are inconsistent and simply irreconcilable. However, courts have increasingly faced this challenge with an eye toward legitimizing reverse engineering of computer software through the ineffectual guise of the statutory fair use defense.

III. ANALYSIS

A. *Ambiguity Sets the Stage for the Final Battle*

It is generally agreed, by opponents and proponents alike, that the present statutory framework is inadequate to address the effects of reverse engineering on copyrighted computer software.[108] On the one hand, attorneys are

---

[107] *See, e.g.*, Rothman, *supra* note 102, at 7 (suggesting the copyright misuse doctrine as a new approach).

[108] *See, e.g.*, Barak D. Jolish, *Rescuing Reverse Engineering*, 14 SANTA CLARA COMPUTER & HIGH TECH. L.J. 509, 512-13 (1998) ("Congress should eliminate the existing state of confusion through legislative action."); Weiner, *supra* note 10, at 12 ("The language of section 117 and the legislative history are both silent on whether this section was meant to encompass reverse engineering of software."); Behrens & Levary, *supra* note 5, at 1 ("Copyright laws that currently govern the development of

concerned about how to advise clients. On the other hand, computer programmers are concerned about the repercussions of various software development methods. Regardless, both are equally worried about the ambiguities within the current governing copyright law [109] The text of the current Copyright Act, together with the interpreting case law, creates a presumption that disassembly is itself an infringement;[110] however, courts have occasionally overcome this presumption by stretching the fair use defense to allow reverse engineering.[111]

Whether analyzing the fair use defense or pondering legislative intent, the law of reverse engineering of computer software is essentially judge-made.[112] While Congress amended the Copyright Act to explicitly include computer software,[113] it failed to explicitly address the issue of reverse engineering as it applies to computer software. The Copyright Act does allow for limited copying by a program acquirer so as to promote the effective intended use of the program, but not to facilitate the development of a competing product,[114] and certainly does not grant such permission to commercial competitors.[115] Moreover, a section of the Copyright Act includes the fair use defense, which limits the exclusivity of the copyright owner's right, but, it is unclear whether this section was intended to encompass reverse engineering.[116] Despite this ambiguity, courts have

---

computer software are still quite ambiguous, with difficulties in their application and debates over the scope of the law.").

[109] *See, e.g.*, Behrens & Levary, *supra* note 5, at 1. Attorneys are affected by uncertainty in giving clients advice on whether certain software development techniques violate copyright laws or carry other legal consequences. *Id.* Similarly, computer software manufacturers need more clarity in this area of law in order to have a better idea of what is and is not allowed when developing new programs. *Id.* at 17.

[110] *See* Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 978, 1016 (1993).

[111] "[J]udicial creation of even a circumscribed reverse engineering privilege under the fair use doctrine seems singularly inappropriate." *Id.* at 1024.

[112] *See* Bayha, *supra* note 2, at 178.

[113] *See supra* Part II.A.1.

[114] Section 117 gives a program acquirer a circumscribed freedom to make a copy or adaptation for internal use, negatively implying that the statute's permission does not extend to commercial competitors. Miller, *supra* note 110, at 1023. *See also supra* text accompanying note 32.

[115] *See supra* note 114 and accompanying text.

[116] *See* Weiner, *supra* note 10, at 12.

increasingly employed the fair use doctrine in order to avoid a finding of copyright infringement.[117]

Congress has not, however, evaded the subject of reverse engineering entirely In providing an allowance for reverse engineering of semiconductors, Congress took a *sui generis* approach to this issue in adopting a specifically tailored provision of the Semiconductor Chip Protection Act ("SCPA").[118] The provision protects infringement, via copying and similarity, provided that the end product is the result of study and analysis and embodies technological improvement.[119]

Opponents of reverse engineering argue that the express right to reverse engineer provided for in the specific provision of the SCPA evinces legislative intent not to allow such activity with respect to other works, such as computer programs.[120] Proponents respond by simply asserting that this argument was rejected by the Ninth Circuit in the *Sega* case.[121] Proponents also argue that cases which invoke the fair use defense stand for the proposition that reverse engineering can be lawful.[122] This argument lacks merit. To begin with, proponents of reverse engineering recognize the clear lack of guidance from Congress as well as the courts.[123] Second, they concede that a strict interpretation of the statutory language could easily lead to a belief that Congress did not intend to allow reverse engineering.[124] Third, they admit the strong possibility that Congress did not intend the provisions of the Copyright Act to apply to reverse engineering.[125] Based on these concessions, the proponents' argument for the legitimization of reverse engineering of computer software is unsubstantiated. Proponents appear to be engaged in a sort of

---

[117] *See supra* Part II.B.

[118] 17 U.S.C. § 906(a).

[119] *See* Bayha, *supra* note 2, at 178.

[120] The argument is that since Congress took the affirmative step of explicitly allowing reverse engineering of semi-conductor chips, it could have easily taken the same step in respect to computer programs. Therefore, since Congress did not provide an express statutory right to reverse engineer computer programs, no such right is to be implied. *See, e.g.*, Miller, *supra* note 110, at 1024.

[121] *See* Bayha, *supra* note 2, at 178.

[122] *See id.* at 185.

[123] *See id.* at 193.

[124] *See* Weiner, *supra* note 10, at 12.

[125] *See id.* at 17.

circular reasoning to: (i) assert that the current legislation is unarguably ambiguous;[126] (ii) recognize that some courts have, therefore, felt it prudent to wait for a legislative response[127] while other courts have simply stretched the fair use defense;[128] and (iii) then support their proposition by relying on courts invoking and interpreting such a clearly ambiguous statute. Proponents support their proposition that reverse engineering can be lawful by relying on a body of case law that they admit is weak and possibly even analytically flawed.[129]

With an inactive legislature and courts hesitant to find infringement in computer software cases, especially in light of the fair use defense, virtually no aspect of a computer program is afforded copyright protection under the current copyright regime. Therefore, programmers are effectively free to disassemble an entire copyrighted program. This "license" to freely reverse engineer is equivalent to giving someone access to the entire copy of a program's source code.[130] Despite the lack of comments and labels, it is nevertheless quite simple to tweak variables and modify the "visual" aspects of the code to mask the "similar expression" yet still "copy" the idea of the program.[131] Given the unarguable fact that current copyright

---

[126] *See* Jolish, *supra* note 108.

[127] *See* Bayha, *supra* note 2, at 194.

[128] It has been argued that in both the *Atari* and *Sega* cases the courts deviated from Congress' intent and design by creating a "wide-angle [disassembly] privilege" despite the telephoto character of the fair use doctrine. *See* Miller, *supra* note 110, at 1015. In both of these landmark cases, the courts accorded weaker copyright protection to computer programs than to other works, explicitly contradicting CONTU's findings and Congress' intention. *Id.* at 1022.

[129] *See* Davis, *supra* note 12, at 160 (stating "courts, lawyers and their clients should be wary of some of the existing case law, which seems wrong").

[130] The argument is simply that an unlimited freedom to disassemble is tantamount to handing over a complete copy of the program's source code. Since it is possible, and quite easy, to simply change variable names and the aesthetic structure of the code itself, programmers are able to successfully copy the protected expression by means of such trivial changes. Provided the new code produces a final product that appears facially dissimilar from the original program, these changes can mask the actual copying of the expression.

[131] While it is relatively simple to circumvent allegations of copyright infringement of computer software, some computer programmers are aware, and cautious, of the risks posed by engaging in reverse engineering. *See, e.g.*, http://www.darbylaw.com/reveng.html (last visited Nov. 6, 2001) (providing procedures to be utilized for reverse engineering of computer code). Intellectual property firms, such as Darby & Darby, are clearly concerned that any reverse engineering comply with strict procedures designed to avoid any possible infringement of copyrighted code.

law is entirely ambiguous and thus provides little, if any, guidance for courts, it is now simply a matter of determining who will prove victorious in the reverse engineering war based on competing premises.

B. *Innovation and Competition Boasting the Battlefront*

The vast majority of legal scholars seem to support the current trend in case law, believing that innovation and competition far outweigh the need to protect computer programs. In arguing their position, proponents of reverse engineering favor public interests over private protection. Their "innovation argument" asserts that the public receives a major benefit when inventors and businesses are encouraged to devise similar products through reverse engineering because it encourages the original inventor to keep ahead of the competition by continuing to improve the original product.[132]

The proponents' "competition" argument claims that reverse engineering encourages healthy competition and price pressures in the market.[133] This view clearly parallels the tendency of courts to favor competition over monopoly [134] The general fear is that prohibiting reverse engineering would give a copyright owner an effective monopoly over all software that would presumably interoperate with the copyrighted work, which monopoly may even extend to corollary markets.[135] Proponents and courts believe that a prohibition on reverse engineering would ultimately "stifle the proliferation of ideas, bounty of products, and hearty competition."[136] Furthermore, at least one proponent argues that the trivial amount of

---

[132] *See, e.g.*, Behrens & Levary, *supra* note 5, at 17; Davis, *supra* note 12, at 147.

[133] *See* authority cited *supra* note 126.

[134] *See* Behrens & Levary, *supra* note 5, at 11.

[135] *See* Bayha, *supra* note 2, at 193. A copyright on certain software, in a body of law that prohibits reverse engineering, would grant the owner a monopoly over all interoperable software, as well as a monopoly that may even extend to the market for computer hardware. *Id.*

[136] Jolish, *supra* note 108, at 512.

reverse engineering done for competitive purposes is most likely done to obtain "obscure little details" as opposed to more general externalities.[137]

Proponents of reverse engineering further sustain their position by proclaiming the relevant needs for reverse engineering within the software industry In accordance with the deemed importance of providing a public benefit, proponents argue that access to ideas is necessary Ideas are not copyrightable subject matter, and reverse engineering provides access to these unprotected ideas.[138] Another need in the software industry for reverse engineering is software compatibility, which is prevented under a strict interpretation of current copyright law [139] Perhaps most importantly, proponents argue that with the rapid evolution of technology, there is a disparate progress in the areas of computer hardware and software.[140] Proponents believe that the disparity between advancements in these convergent industries could be reduced through the use of reverse engineering.[141]

In addition to the needs within society and within the industry, public policy will inevitably have a significant impact on the outcome of the reverse engineering war. Laws that are interpreted to prevent the public from using ideas and information in the public domain violates public policy, which demands that information and ideas are kept available for all to use.[142] In this sense, proponents argue that a prohibition on reverse engineering would violate public policy by affording a copyrighted computer program monopolistic-like protection.[143]

---

[137] Johnson-Laird, *supra* note 7, at 352.

[138] *See* Weiner, *supra* note 10, at 6.

[139] *See id.*

[140] *See id.* at 7.

[141] *See, e.g.*, Weiner, *supra* note 10, at 6. Weiner uses the Intel Pentium Pro CPU as an illustration of this point, arguing that while the CPU is optimized for 32-bit applications, very little 32-bit application software is on the market. By allowing software manufacturers to reverse engineer the limited 32-bit application software available, for the purpose of achieving interoperability and compatibility, the utility of the hardware advancement made by the Intel CPU would increase.

[142] See Justice O'Connor's discussion of conflicting patent and copyright protections in the landmark case of *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.*, 489 U.S. 141 (1989).

[143] *See* Davis, *supra* note 12, at 155. Despite this seemingly sound argument,

C. *Protection Still Able to Power Ahead*

Despite the strength of the proponents' arguments for favoring innovation and competition over protection, it is not clear that a statutory exception for disassembly would in fact benefit society.[144] Proponents, as well as courts, espouse the objective of copyright law as one to promote competition and innovation in the marketplace.[145] However, the Copyright Act does not suggest that every aspect of a work be made available to the public in order to qualify for copyright protection.[146] As a matter of fact, the disassembly exception that proponents advocate would deny owners the exclusive rights they are statutorily entitled to: the right to authorize, or refuse to authorize the reproduction, adaptation, and translation of their copyrighted works.[147] Permitting the copyright owner to control the intermediate copying incident to reverse engineering increases protection against infringing final products,[148] in addition to making disclosure of "functional" source code self-regulating.[149] More importantly, allowing disassembly actually

---

reliance on the CONTU report does not provide strong support. The report states simply that programmers are free to "read" copyrighted programs; this freedom is not challenged by opponents. *See also supra* note 32 and accompanying text.

[144] "[T]here is little evidence that more convenient access to the ideas embedded in computer code would be of greater benefit to society than leaving the copyright incentives unimpaired by a [disassembly] exception." Miller, *supra* note 110, at 1029.

[145] The power granted to Congress by the Constitution, and the stated objective of copyright law, is "to promote the Progress of Science and useful Arts," by allowing society to benefit from the availability of creative works. U.S. CONST. art. I, § 8, cl. 8.

[146] Copyright is intended to benefit society from the availability of creative works regardless of whether literal expressions or underlying ideas of those protected works are directly available to the public. Miller, *supra* note 110, at 1029. However, the ideas underlying computer programs are usually accessible by simply studying their operation and documentation, although accessing these ideas may require some effort and the cumbersome task of reading machine code directly. *See id.*

[147] *See id.* at 1027.

[148] *See id.* at 1022.

[149] If source code for an unprotected process or function is obtainable by any other legal means, such options should be explored before resorting to reverse engineering. *See* Behrens & Levary, *supra* note 5, at 15; *supra* Part II.A.1. (discussing copyright protection for literal versus nonliteral, or functional, elements of computer programs). The idea here is that if copyright owners are given the chance to make certain code accessible freely or upon request, copyright owners retain more control over their statutory right and disclosure is more self-regulating. If the copyright owner

stifles innovation. By granting freedom to wholly disassemble another's copyrighted software, computer programmers will no longer have any incentive to produce an innovative or creative expression in the first place.[150] An effective "license" to disassemble penalizes the creative effort of the original programmer by freezing or substantially impeding human innovation and technological growth,[151] in addition to amplifying the threat of software piracy [152]

Allowing protection of copyrighted computer programs encourages innovation, and results in greater protection for the software industry as a whole. Additionally, such protection is also imperative to realizing the potential within the software industry At least one oft-quoted scholar has argued that permitting disassembly allows a second programmer to "create a market substitute and reap the benefits of a successful program after others have incurred the risk and expense of its development."[153] When the goal of engaging in disassembly is to replace the original program on the market, benefit would inure only to a small number of companies that have "staked their future on copying their competitor's product," while substantially harming the overwhelming majority of the market.[154] Furthermore, "traces of copying can [easily] be disguised." [155] Since the copyright owner faces the undue burden of discerning and proving infringement, programmers

---

believes the relevant portion of code is protected expression and refuses to disclose it, exercising the right to refuse authorization acts as a warning to subsequent programmers that any attempt to reverse engineer may be an infringement.

[150] *See* Miller, *supra* note 110, at 1027.

[151] *See id.* at 1034.

[152] The primary detriment of allowing reverse engineering in the software industry is the fear of piracy, as well as the potential decrease in initial innovation. *See* Weiner, *supra* note 10, at 9.

[153] Miller, *supra* note 110, at 1026. Some purport that it is far too simple to make slight modifications to an original program through reverse engineering and then market the amended version as a new software product; others argue that reverse engineering is extremely difficult, time consuming, and demanding of the skill and experience of a programmer. *Compare* Behrens & Levary, *supra* note 5, at 5, *with* Johnson-Laird, *supra* note 7, at 346.

[154] Johnson-Laird, *supra* note 7, at 351.

[155] *See* Miller, *supra* note 110, at 1027.

who disassemble are encouraged to engage in hunting expeditions through copyrighted programs and to limit the amount of code they copy for use in their final products.[156]

The argument against reverse engineering is further bolstered by the indispensable perspective of those skilled in the field—computer programmers. Reverse engineering entails a great expenditure of time, skill, expertise, and money Given the time, effort, and expense required to reverse engineer an entire program, it would be much easier and more economical to develop entirely new software from scratch.[157] One computer programmer, in particular, states that reverse engineering is a fundamental part of normal software development and argues that programmers would much rather be challenged to design their own code because that is precisely the kind of intellectual creativity they enjoy [158] If reverse engineering is so overly demanding of programmers who would rather design their own code anyway, then to engage in reverse engineering of copyrighted code might be seen as a futile effort; it is neither quicker nor less expensive, and inevitably risks infringing on copyrighted software.

Finally, public policy plays a significant role in substantiating the opponents' position. While courts have tended to favor the views of proponents while the legislature has remained silent, in general, American interests tend to favor strong intellectual property protection for computer software, including a prohibition on disassembly [159] In the debate over the propriety of reverse engineering, commentators and policymakers have taken some strong stances.[160] The United States Trade Representative ("USTR") has taken the position that as a matter of U.S. international policy, reverse

---

[156] *See id.* at 1018 (discussing the *Sega* case and the court's focus on the final product). It should be noted that although the author shares much of Miller's reasoning and logic, the conclusions that each arrive at are different. Both agree that there should not be a disassembly exception, but since there is not an explicit one, Miller does not feel the need for legislative action since courts are thus free to interpret the Copyright Act. The author does not support Miller's conclusion, in part, because Miller concludes by stating that most judicial decisions seem correct, yet concedes throughout that the *Sega* and *Atari* courts got it wrong.

[157] *See* Johnson-Laird, *supra* note 7, at 348; Davis, *supra* note 12, at 151.

[158] *See* Johnson-Laird, *supra* note 7, at 352.

[159] *See* Bayha, *supra* note 2, at 188.

[160] *See id.* at 178.

engineering is anticompetitive and unlawful under current domestic law [161] In response to concerns over the development and final resolution of the Software Directive in the late 1980s,[162] the United States has further articulated its position. The United States expressed concern over the possibility of amending the Japanese Copyright Act to expressly allow disassembly and reverse engineering of computer software.[163] An advisory council to the Cultural Affairs Agency, including members from the government, academia, and industry, was appointed to review the Japanese Copyright Act.[164] Immediately, the council's activities incited much alarm and trepidation within the industry This overwhelming response from U.S. business interests resulted in a formal request from the Secretary of Commerce and the USTR that "no action be taken without dialogue with interested U.S. parties."[165] Less than a year later the council's report was released, making no recommendation on the reverse engineering proposal, instead labeling such a proposal "premature."[166] In response to the council's report, the Japanese government immediately dropped its pursuit of the reverse engineering issue.[167]

D *So Who Wins The War?*

With innovation and competition battling the need for protection, the question remains who wins the war. First, Congress must act since courts are devoid of any meaningful guidance.[168] Second, proponents of reverse engineering argue that society and the software industry would be better served by providing a disassembly exception to copyright protection

---

[161] *See id.* at 179.

[162] The USTR, in taking its position that disassembly is illegal in the United States, had argued that it should, likewise, not be allowed under the laws of the European Union. *See id.* at 188; *supra* Part II.A.2.

[163] *See* Bayha, *supra* note 2, at 190.

[164] *See id.*

[165] *Id.*

[166] *Id.* In the year between the appointment of the council and the rendering of its report, the Japanese proposal caused continued strife in trade negotiations between the United States and Japan. *See id.*

[167] *See* Bayha, *supra* note 2, at 190.

[168] *See supra* Part III.A.

that will promote innovation and competition in the marketplace.[169] Third, opponents of reverse engineering advocate that granting the freedom to disassemble, as proponents suggest, would actually defeat the incentive that disassembly is purported to provide.[170] To promote the progress of science and the useful arts, the basic premise of copyright law,[171] programmers must be confident that their initial innovation will receive some protection. An unrestricted freedom to disassemble is too broad. Therefore, the necessary logical conclusion is for the legislature to place explicit limits on the right to reverse engineer copyrighted computer programs. The need to further protect the intellectual property of the computer software industry demands that the legislature more clearly define and delineate the role that reverse engineering plays in the realm of copyright law

IV PROPOSAL

A. *News Flash—Reverse Engineering War Ends in Peace!*

1. Terms of the Proposal

This Note proposes an end to the reverse engineering war—to amend current copyright law to explicitly address the reverse engineering issue as it applies to computer software. The Copyright Act should be amended to: (i) expressly allow black box analysis, and (ii) limit disassembly solely to the confines of the clean room process. This limited right to disassemble a copyrighted computer program is permitted only where access to critical yet unprotected ideas and elements is not available by any other legal means. In addition to permitting disassembly only as a means of last resort, the final product must nevertheless satisfy the "substantial similarity

---

[169] *See supra* Part III.B.

[170] *See supra* Part III.C.

[171] *See supra* note 145 and accompanying text.

test" as expressed by both the *E.F Johnson v. Uniden*[172] and *NEC Corp. v. Intel Corp.*[173] courts. Such an amendment might read:

§ 117A. Limitations on exclusive rights: Reverse engineering of computer programs

Notwithstanding the provisions of section 106, it is not an infringement for the owner of a copy of a computer program to engage in reverse engineering as a means to make a software program compatible with pre-existing software to the extent that:

(a) the owner may, without the authorization of the rightholder, observe, study, or test the functioning of the program in order to determine the unprotected ideas and functions that underlie any element of the program if done while performing any of the lawful acts of loading, displaying, running, transmitting or storing the program, which process is referred to as "black box analysis;" or

(b) the owner may, without the authorization of the rightholder, gain access to necessary portions of the program through the process of translating object code to source code, provided:

(1) such access is unavailable from other sources such as program manuals, flow charts, interface specifications or other technical documentations;

(2) such portions of disassembled code are limited to those that contain unprotected ideas and functions; and

(3) that the act of disassembling occurs strictly under "clean" conditions where:

(i) the first programmer, or team, develops a set of high level specifications or criteria necessary to create the final product, achieved by engaging in any lawful activities, including black box analysis;

(ii) the resulting set of specifications or criteria is free from any of the protected expression of the original program;

(iii) such set of specifications or criteria is then received by a second programmer or team that has never seen or been exposed to the original program, and programs new code based exclusively on such set of specifications or criteria; and

(iv) the first and second programmer are not permitted to communicate directly at any time during the development of the program.

---

[172] 623 F Supp. 1485 (D. Minn. 1985).

[173] 10 U.S.P.Q.2d 1177 (N.D. Cal. 1989).

These provisions would serve as an affirmative defense; the burden of proof would be on the alleged infringer to rebut the copyright owner's prima facie case of infringement. If the defendant asserts an affirmative defense based on the use of black box analysis, any similarities could be attributed to specific compatibility and hardware constraints. For an affirmative defense based on the clean room process, "paper trail" evidence could be used to prove the legitimacy of the reverse engineered product.[174] Since programmers would be allowed to engage only in black box analysis or disassembly via the clean room process under the new statutory framework, the potential for infringing software is greatly diminished.

Presuming that programmers will not engage in illegal methods of reverse engineering once their rights are delineated, the legislature would be able to seal the proposal/amendment. Each side gains and loses a little of its previous freedoms, however, industry and society will equally benefit.[175] The most significant trade-off occurs within the industry While programmers who may at some point be copyright owners will no longer be discouraged from creating more sophisticated software since they are afforded somewhat more protection, those same programmers who may become second programmers at some other point in the future are still allowed enough access to original programs to encourage innovation. By providing equilibrium within the industry, proponents and opponents are both able to attain their goals. Providing protection allows innovation and competition to thrive in an industry where private protection essentially transpires into a public benefit.

---

[174] The clean room process still poses the threat of potential infringement, given that disassembly will generally be involved, since the process would require the making of intermediate copies. However, the technique may be effective if disassembly is permitted on other grounds, such as within the confines of the proposed amendment. If the initial disassembly is not legal though, use of the clean room process alone will not legalize the final product. *See* Miller, *supra* note 110, at 1025.

[175] However, given that these freedoms emanated from statutory ambiguity, they were never definitive rights.

2. Closer to the EU Software Directive?

The overwhelming concern expressed by the United States, in light of the Software Directive and the proposed amendments to the Japanese Copyright Act, strongly suggests that any amendment to current copyright law reflect those concerns.[176] The amendment proposed in this Note balances the views and fears posed by both sides. By expressly permitting reverse engineering in the form of black box analysis, and strictly limiting acts of disassembly to the confines of the clean room process, both as a means to achieve interoperability with pre-existing software, the proposal addresses reverse engineering of computer software in a way that provides protection yet encourages innovation and competition.

Black box analysis occurs through a process where the computer programmer never looks at the source or object code of the program. Since any similarities in code will not be the result of copying from the original code, this form of reverse engineering has generally been deemed permissible by both opponents and proponents alike.[177] Similarly, the clean room process will produce code that is not "substantially similar" to the original copyrighted code, given that the programmers who developed the new code never had access to the original code.[178] The benefit of allowing disassembly only as a function of the clean room process is that since there is no substantial similarity, the new code presumably will not be an infringing work because access and substantial similarity could not be established as required for proof of copyright infringement.[179]

---

[176] *See supra* Part III.C.

[177] *See* Bayha, *supra* note 2, at 179 ("[black box analysis] generally has been deemed permissible by both protectionist and non-protectionist schools alike."); Davis, *supra* note 12, at 146 ("There should be no serious question that [black box analysis] is legal, as it has been for more than a hundred years, and it does not involve any copying of the source or object code of the original device.").

[178] *See* Bayha, *supra* note 2, at 181. In the absence of verbatim copying, as would be absent in the clean room process, a copyright owner may show infringement only by showing that the alleged infringer had access to the original work and that the two works are substantially similar. *Atari*, 24 U.S.P.Q.2d at 1024.

[179] *See* Bayha, *supra* note 2, at 181.

Furthermore, the clean room process appears to have been expressly approved in both the *Computer Assoc. Int'l v. Altai, Inc.*[180] and *NEC Corp. v. Intel Corp.*[181] cases.

While black box analysis is unarguably permissible,[182] there is concern over the clean room process because it still involves intermediate copying of the copyrighted code, despite the lack of access and substantial similarity [183] Like the EU, which struggled with this very issue in the consideration of its Software Directive, the U.S. legislature must determine how best to protect copyrighted computer software while allowing enough access to achieve interoperability Clearly, if black box analysis, together with access to any relevant technological documentation, fails to provide the programmer with sufficient information, a viable alternative would be to disassemble necessary portions of the original code. To this extent, the Software Directive provides a noteworthy model for structuring amendments to the Copyright Act. However, given the intense opposition to allowing a blanket exception for disassembly, as attempted by the Japanese government, the Software Directive's limitations on disassembly[184] would be further improved by limiting permissible disassembly to the clean room process. Since a broad freedom to disassemble is so highly objectionable, the clean room process provides additional protection from the possibility of infringement. By moving closer to the EU Software Directive, but providing an additional safety net for programmers, the proposed amendment to the Copyright Act strikes the proper balance between competing concerns.

B. *Aftermath of the War: Economic and Societal Benefits of Proposal*

The economic model of intellectual property protection requires a level of societal protection that will maximize

---

[180] 982 F.2d 693 (2d Cir. 1992), *aff'g* 775 F. Supp. 544 (E.D.N.Y. 1991).

[181] 10 U.S.P.Q.2d 1177 (N.D. Cal. 1989).

[182] *See* sources cited *supra* note 177.

[183] *See supra* note 177; Bayha, *supra* note 2, at 181.

[184] *See supra* Part II.A.2.

societal benefits while minimizing societal costs.[185] While a monopoly is generally less efficient than a market economy, intellectual property is an unusual commodity, and a successful economic model can be achieved through a grant of certain, limited monopoly rights to the creator of intellectual property.[186] By granting computer software monopoly protection for a limited time through the copyright regime, programmers are provided a monopoly incentive that ultimately produces a greater quantity of ideas and original inventions. In addition to quantitative improvements, qualitative improvements are attainable through stronger protection.[187] In vast industries, such as the software industry, where the research and development costs are high and the production costs are low, few companies are willing to invest in the development of revolutionary inventions if competitors can easily copy them.[188] By amending current copyright law to provide for greater protection, but not prohibitive overprotection, society and the software industry will benefit from a significant improvement to the quantity and quality of software developments.

The economic impact of allowing a reverse engineering exception would be devastating to the software industry, given the immense discrepancy between the cost of creating software versus the cost of merely duplicating it. Although the cost of engaging in reverse engineering can be substantial for large programming projects,[189] original programs are increasingly expensive to develop.[190] On the one hand, the creation of original computer software requires large teams of programmers to commit extensive amounts of time and the actual software company to expend enormous amounts of money. On the other hand, once fully developed, the reproduction costs of the software are insignificant in

---

[185] *See* Lawrence D. Graham & Richard O. Zerbe, Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure*, 22 RUTGERS COMPUTER & TECH. L.J. 61, 71 (1996), for a complete economic analysis of the effects of reverse engineering on computer software.

[186] *See id.*

[187] *See id.* at 72.

[188] *See id.*

[189] *See* Davis, *supra* note 12, at 151.

[190] *See* Graham & Zerbe, *supra* note 185, at 68.

comparison. Therefore, computer software is a highly attractive target for reverse engineering. If an exception were permitted, a programmer engaged in disassembly would be able to reproduce a competitor's entire program and appropriate in a single procedure what could represent years of creative effort and investment by the copyright owner.[191] Due to the unfair economic advantage that reverse engineering provides second comers, and the threat that this imposes on the software industry, the economic benefits of this Note's proposal further support its efficacy [192]

CONCLUSION

This Note has analyzed the impact of reverse engineering of computer programs within the software industry under current copyright law The analysis has demonstrated the difficulty that courts have encountered in walking the statutory fine line between the need for protection and the need to encourage innovation and competition. In critiquing the current trend of case law, which stretches an inadequate statutory framework beyond its intended scope, the analysis illustrates that an amendment to the Copyright Act is imperative to provide the courts with any meaningful guidance. In particular, this Note has proposed an amendment to expressly legitimize the practice of black box analysis and a limited right to disassemble by placing these exceptions squarely within the provisions of the Act. The proposed amendments are carefully tailored to provide additional protection to copyrighted computer software without impeding

---

[191] *See* Miller, *supra* note 110, at 1026.

[192] Notwithstanding the economic incentive to reverse engineer software, market forces frequently provide protection against products developed through reverse engineering. *See* Graham & Zerbe, *supra* note 185, at 68. Lead time, for instance, gives the original developer advantages in brand recognition, as well as brand loyalty if the original program is on the market for a significant amount of time prior to the release of any similar, subsequent programs. *See id.* at 69. Moreover, the shelf life for software products is relatively short and the original developer may have upgraded its original version before the second developer's product has a chance to hit the market. *See id.* However, there is also an economic disincentive to engaging in reverse engineering; given the likelihood that original software may contain errors, reverse engineering such software may be more costly than independent development. *See id.* at 66.

innovation or competition. The amendments also provide protection to effectively encourage innovation and competition and permit access to the program's unprotected ideas and functions. The public and software industry can equally benefit from the creation of interoperable software through a provision that explicitly addresses reverse engineering. Through a straightforward application of the amendments, the future of reverse engineering of computer software is no longer in flux, and the legislature can finally put an end to the reverse engineering war.

Barbara J. Vining†

† Bachelor of Arts in Computer Science (1999), New York University; candidate for degree of Juris Doctor, Brooklyn Law School (2002).