14 Tex. Intell. Prop. L.J. 59

Texas Intellectual Property Law Journal
Winter, 2005

Article

A SUM GREATER THAN ITS PARTS?: COPYRIGHT PROTECTION FOR APPLICATION PROGRAM INTERFACES

Efthimios Parasidis[a1]

Table of Contents

*60 Introduction

The exponential growth exhibited in the technology sector worldwide has provided businesses and consumers with a myriad of computer programs and computer-based devices that have significantly altered and improved our everyday lives. Whereas the copyright laws of the United States have attempted to keep pace with these rapid technological advances, in certain instances, such has not been the case.[1] One such area involves the extent of copyright protection for application program interfaces ("APIs").[2] Whereas APIs have been implicated as important segments of major international lawsuits involving large, multi-national corporations such as Microsoft,[3] few scholars have addressed whether, and to what extent, APIs are protected under the copyright laws.[4]

***61** In this essay, I will discuss whether APIs are protected under the copyright laws of the United States and, for those instances where APIs are protected, the level of protection afforded. Additionally, I will consider the defense of fair use as it relates to copyright infringement claims regarding APIs.

I. What is an API?

The term API includes "[s]oftware that an application program uses to request and carry out lower-level services performed by the computer's . . . operating system."[5] In particular, APIs include header files, data structure information, call ***62** mapping information, function prototypes, variables, parameters and constants,[6] data and file format specifications, function calls,[7] screen displays, function name calls, and other information[8] required to develop application programs which interoperate with an operating system.[9] Essentially, an API ensures that all applications are consistent with the underlying operating system and have a similar user interface with that system.[10]

***63** Standardization of APIs at various layers of a software program provides a uniform way to write applications.[11] As such, APIs provide the standard environment, including tools, protocols, and other routines, in which programs can be written.[12] APIs are a set of standard software interrupts, calls, and data formats that application programs use to initiate contact with network services, mainframe communications programs, telephone equipment or program-to-program communications.[13]

***64** Essentially, APIs are specifications which facilitate the communication between various aspects of a computer program. Specifically, APIs are the exact "method prescribed by a computer operating system or by an application program by which a programmer writing an application program can make requests of the operating system or another application."[14]

II. How APIs May Be Copied

Understanding the role that APIs play in a computer program, and how they may be copied, is best illustrated through a paradigm. Assume that Company A is the manufacturer of a handheld personal data assistant ("PDA"). In addition to manufacturing the actual PDA device, Company A creates software applications that run on its device. Company B, on the other hand, is not in the business of creating devices, but rather is a software developer. Company B sees great potential in the device created by Company A, and determines that it can produce software compatible with Company A's device that is far superior to the software produced by Company A. Assuming that Company A does not grant Company B any rights to use the software or device created by Company A, the most common and efficient method by which Company B will be able to design software that will be compatible with Company A's device is through decompilation of Company A's software.

***65** Decompilation is a form of reverse engineering.[15] Under the copyright laws, Company B is permitted to decompile Company A's software program and create its own software based on the information obtained from the decompilation process.[16] Decompilation techniques were initially used in the 1960s to aid in the migration of computer programs from one platform to another.[17] To decompile is to convert executable program code, sometimes referred to as object code, into some form of higher-level programming language so that the code can be read by a human.[18] The tool that accomplishes this conversion is called a decompiler.[19]

***66** In addition to reverse engineering for the purpose of developing compatible software, there are a number of different reasons for decompilation, such as understanding the mechanics of a program, recovering the source code for purposes of archiving or updating a program, finding viruses, debugging programs and translating obsolete code.[20] As a result of the mechanics of reverse engineering and computer programming, intermediate copies of APIs are created during the decompilation process.[21] Accordingly, when a computer program is decompiled without the permission of the copyright holder of the program, a claim for copyright infringement may be introduced.[22]

*67 III. Copyright Protection for APIs

The copyright laws of the United States were enacted by Congress under its Constitutional mandate to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[23] The copyright laws are codified in the Copyright Act of 1976, 17 U.S.C. §§ 101.[24] By amendment in 1980, the Copyright Act specifically includes computer programs as proper subjects of copyright protection.[25]

As with all subjects of copyright, computer programs are protectable only to the extent that they incorporate authorship in a programmer's original expression of ideas, as distinguished from the ideas themselves.[26]

Copyrighted software typically contains both protected expression and unprotected or functional elements.[27] Whereas the copyright laws of the United States generally do not protect functional aspects of works, processes, systems or methods ***68** of operation, computer programs are inherently functional works and the most important and creative aspects of computer programs are often the procedures, processes, systems or methods of operation.[28] With respect to APIs, although it is an incorrect statement of law that interface specifications are not copyrightable as a matter of law, external considerations, such as compatibility, often negate a finding of infringement.[29] Various courts have held that such program interfaces are functional elements of a software program, and thus may not be deemed protected expression under the Copyright laws, while other courts have determined that copyright protection for APIs is warranted.[30] As such, a careful examination of the case law is required in order to understand the scope and extent of copyright protection for APIs.

A. Cases Indicating Copyright Protection for APIs is Not Warranted

A number of federal courts have held that copyright protection for APIs is not warranted.[31] One theory, adopted by the Eleventh Circuit, reasons that, to the extent that copying of APIs is required for purposes of compatibility, copyright protection ***69** is not appropriate.[32] Similarly, other courts have held that, where the structure of a program's APIs is dictated by external factors, or where the structure of a program's APIs merges with the underlying function of the program itself, a claim of copyright infringement will not be upheld.[33]

In Bateman v. Mnemonics, Inc., Mnemonics created an application program that was designed to be interoperable with Bateman's system.[34] As with all programs that seek to interoperate with another system, Mnemonics's interface specifications had to be functionally identical to those found in Bateman's program.[35] In particular, Mnemonics utilized system calls found in Bateman's software in order to allow Mnemonics's software to effectively communicate with Bateman's system.[36] In this respect, the Eleventh Circuit was faced with the question of whether interface specifications between an operating system and applications written to run under that operating system are protectable under the copyright laws.[37] Although the court did not issue a definitive answer on the issue of copyright protection for Bateman's APIs, it indicated that copying for purposes of compatibility may negate copyright liability, since compatibility-required elements are not copyrightable.[38] In this respect, Bateman stands for the proposition that, to the extent that APIs are necessary for compatibility, they may not be the basis for a copyright infringement claim.[39]

Analogous to the facts surrounding Bateman are those presented to the court in Mitel Inc. v. Iqtel, Inc.[40] In Mitel, a federal district court in Colorado held that command codes used to program a manufacturer's telephone call controller were ***70** not copyrightable.[41] The command codes in Mitel involved three and four digit numbers or letters that engaged a particular function of the call controller, such as automatic redial or speed dialing.[42] Although the command codes were not directly accessed by the telephone companies that bought the devices or by the telephone service customers, they were used by technicians who programmed the controllers in the process of installation.[43] Once the devices were coded and installed, telephone service customers were able to access the functionalities that the call controllers enabled, though the telephone customers did not themselves enter the call controller codes.[44]

Iqtel did not dispute that it had copied Mitel's three and four digit command codes for use in its telephone call controller.[45] Rather, Iqtel maintained that, for its call controller to be competitive, it had to be compatible with Mitel's call controller, since Mitel had 75% to 90% of the call controller market and technicians expressed reluctance to learn new command codes.[46] Mitel responded by arguing that its codes constituted protected expression because they represented a creatively chosen set of alpha-numeric codes.[47] Iqtel countered by characterizing the command codes as an unprotectable method of operation under 17 U.S.C. § 102(b), and claimed that the scenes a faire and fair use doctrines also precluded a claim for copyright infringement.[48] The court agreed with Iqtel, holding that the command codes were not subject to copyright protection because they were "a procedure, process, system, and method of operation."[49]

***71** The command codes at issue in Mitel may be analogized to APIs.[50] Once programmed, the call controller codes were continually and automatically invoked by the system.[51] Further, with respect to functionality, the command codes were the only codes that were recognized by the call controllers.[52] In this respect, the command codes are similar to technical interface elements, such as header names, in that both are comprised of sets of short terms that technical personnel must program into a device or program in order to invoke an underlying functionality.[53] Thus, whereas the Bateman court held that APIs that are necessary for compatibility purposes may not be the basis for a copyright infringement claim, the court's ruling in Mitel indicates that the copyright laws do not afford APIs protection in circumstances where the APIs are properly characterized as a method of operation.[54]

Building on Bateman and Mitel is Baystate Technologies, Inc. v. Bentley Systems, Inc., which provides the strongest argument against copyright protection for APIs.[55] In Baystate, the court held that the data structure names and the organization of data structures of a computer program were not protected by the copyright laws.[56] Baystate held copyrights for a mechanical computer aided design ("CAD") program.[57] By utilizing computers as drafting devices, CAD programs "enable architects, engineers and other design professionals to design and alter the design of buildings, mechanical devices and electronic equipment."[58]

***72** Bentley Systems, a competitor in the market for CAD programs, sought to create a translator program which was to employ certain elements from Baystate's software.[59] The district court identified the copied elements of Baystate's software as the "names of the so-called data structures . . . and the organization of the files within the data structures," including, "more specifically, the words and abbreviations used to describe the files contained within the data structures and the data structures themselves."[60] These structures were found in the header files of Baystate's program.[61]

The court noted that the Copyright Act defines a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[62] In this respect, the court reasoned, the overall computer program comprising the allegedly infringed work was copyright protected.[63] However, since the data structures at issue in the case did not bring about any result on their own, the court indicated, "they were copyright protected, if at all, only as a part of the whole computer program."[64] With respect to the APIs, the court held that the data file names were unprotected because the APIs merged with the underlying idea or function of the program, and that, based on the scenes a faire doctrine, copyright protection was not warranted for the organization of the data files.[65]

The merger doctrine bars protection of the "data structure names or, more specifically, the words and abbreviations used to describe the files contained within the data structures and the data structures themselves."[66] According to the Baystate court, whereas the data structure names were independently created and were original expression, there was evidence that "the name of a file is typically related to its function."[67] As the court reasoned, under the scenes a faire doctrine, copyright protection does not extend to elements of a computer program that have been dictated by external factors.[68] For computer software, the court noted, such "external factors ***73** include, among other things, compatibility requirements and industry-wide programming practices."[69] In Baystate, the court concluded that "the selection and organization of the elements in the data files is dictated mainly by external factors."[70] As the court indicated:

> [T]he product being developed is a data translator that is designed to "read" the data files of CADKEY. The process of "reading" the CADKEY data files requires that the elements contained within the data structures of the Translator be organized in the same manner as the elements in the data structures of CADKEY.[71]

Thus, the court held that the inability of the translator to function unless it was compatible means that the structures devised by the original author were not protected by the copyright laws.[72] The court bolstered its conclusion that the structures were not copyrightable with further references to the needs of the copier: "[T]he organization of [file] names is, at least partly, a function of efficiency," because "the names and arrangement of those names serve a functional and necessary purpose in the code of a data translator."[73]

Although the bulk of the Baystate court's copyright analysis is devoted to the issue of protectability, the court briefly considered whether the data structures had been infringed, assuming for the sake of argument that the data structures were protectable.[74] In other words, the court considered whether the copying of data structures represented extensive enough copying to render the two works substantially similar.[75] The court concluded that they were not sufficiently similar because data structures represented "neither a substantial portion nor a significant aspect of the whole copyrighted work."[76] The expert testimony presented in the case demonstrated that, "although data structures are generally a necessary component of a computer program for organizational and efficiency purposes, the original naming of the data structures takes very little of the total time or creative genius necessary to develop a program."[77] Furthermore, the court noted that the data structures were not, by themselves, executable.[78] Thus, although the importance of a program ***74** component is not strictly a function of quantity, the evidence in Baystate demonstrated that the data structures were only a fraction of the total CADKEY program, and that copying of the data structures would not have made the two programs substantially similar.[79]

Taken together, Bateman, Mitel and Baystate provide persuasive arguments against copyright protection for APIs.[80] In particular, the cases stand for the position that, to the extent that copying of APIs is necessary for purposes of compatibility, a claim of copyright infringement will not be upheld.[81] Moreover, the cases indicate that the merger and scenes a faire doctrines are such that a claim of copyright protection for APIs, where the APIs merge with the underlying program or are dictated by external forces, is not defensible.[82] In other words, the copyright laws do not encompass situations where the structure of a program's APIs is dictated by external factors or where the structure of the APIs merges with the underlying function of the program itself.[83]

Notwithstanding the fact that APIs may not be deemed protected expression for a particular computer program, however, a company may not make a verbatim copy of copyrighted software.[84] As such, where the commands and structure of a computer program are highly functional, a virtually identical copy of the program may be a basis for a copyright infringement claim.[85]

B. Cases Indicating Copyright Protection for APIs is Warranted

In light of the analyses behind the decisions in Baystate, Bateman and Mitel, it would seem difficult to imagine that a company could successfully argue that the APIs of a particular computer program constitute protectable expression under the ***75** copyright laws.[86] To the contrary, this position is significantly strengthened by decisions of several federal district and circuit courts.[87]

In granting plaintiff's motion for a preliminary injunction, the court in Control Data Systems, Inc. v. Infoware, Inc. held that defendant's emulator program likely infringed copyrights in the technical interface and source code of Control Data's network operating system ("NOS").[88] Infoware's program was designed to permit application software that was originally written for Control Data's Cyber computers and operating system to be used with the hardware of other manufacturers.[89] The alleged similarities between the NOS operating system and defendant's emulator software included: (1) 2,000 lines of copied NOS source code; (2) the NOS input and output formats; (3) NOS file layouts; (4) NOS source code parameters; and (5) NOS commands.[90]

Since much of the evidence presented in Control Data Systems was produced pursuant to a protective order, the opinion of the court sets forth only a limited description of the facts.[91] In this respect, the factual basis of the court's decision is not entirely clear.[92] Nevertheless, from the facts available in the public record, it appears that the district court primarily relied on the source code copying, though the court did not discuss or analyze whether the other elements constituted protectable expression under the copyright laws.[93] In particular, the discussion in Control Data Systems focuses on whether the merger doctrine precludes a claim of copyright infringement, such that Infoware was required to create a NOS-compatible operating system in order to compete in the relevant market.[94]

***76** The court held that, based on the facts of the case, the merger doctrine did not preclude a claim for copyright infringement by Control Data Systems, since the elements at issue were not dictated by the underlying computer code.[95] In other words, the program created by Control Data did not represent the only way to make a Cyber-compatible operating system.[96] In this respect, the court's discussion in Control Data Systems may be reasonably construed to extend copyright protection for APIs to instances where the APIs are not dictated by the need for interoperability with an underlying computer program.[97]

In addition to Control Data Systems, CMAX/Cleveland, Inc. v. UCR, Inc. provides support for the proposition that APIs may be afforded protection under the copyright laws.[98] Plaintiff CMAX owned the copyright for a computer program called "RMAX," which was used to input, store, process and retrieve information related to the rent-to-own furniture and appliance business.[99] In an effort to avoid CMAX's license fees, UCR, who had originally licensed CMAX's software, set out to develop its own program that would perform the same functions as RMAX and be compatible with files and records previously created using RMAX.[100] To this end, UCR developed a program of identical design to RMAX for its own use.[101] In order to ensure compatibility with its existing RMAX data files, UCR studied the file structure and file names of the RMAX program and replicated them in its own program.[102] In addition, UCR replicated many of RMAX's screens and reports.[103]

The court ruled that the file layouts or structures, record layouts, file names, naming conventions, transaction codes, screens and reports of the RMAX program constituted protectable expression because they were not dictated by industry standards, by efficiency, or by the need for the program to interact with the central host ***77** computer with which it was designed to operate.[104] Indeed, although the court was not entirely clear about which elements constituted aspects of the user interface or the technical interface, some of the elements which the court held were protectable expression are properly characterized as APIs.[105]

For instance, the file structures in the case included file names, sequences of fields, and field names or field definitions.[106] The court found that these structures for inputting data were not dictated by market forces and that it was not functionally required that they follow a particular order.[107] The court further held that, since such file structures collect and organize information entered by the user, they are not merely blank forms, and that there is protectable expression in the selection and arrangement of the field definitions.[108] In this respect, the field names and sequences function as APIs.[109]

In addition, the CMAX court held that the screens, reports and transaction codes constituted protected expression.[110] With respect to the screens and reports, since the field definition sequence in the screens and reports was not dictated by the industry, and since the data fields could have been arranged in any number of ways, they represented protected expression.[111] Further, whereas screens and reports are user interface elements, in that the user sees and works with these elements when entering data and reviewing results as part of the stored files, these elements become a technical interface insofar as they are used and can be read by other programs.[112] Similarly, with respect to the transaction codes, the court determined that, since they were a major element of the program's design and were not dictated by efficiency or industry demands, they were protected by the program's copyrights.[113] As defined by the court, transaction codes are equivalent to APIs: "A transaction code is a randomly selected, alphanumeric sequence of characters that indicates to the computer what steps should be executed in a given situation or 'transaction' when the code is transmitted."[114] However, it is apparently the end ***78** user of the program that directly employs the transaction codes.[115] In this respect, the transaction codes are similar to the Mitel telephone system command codes in that the transaction codes are analogous to header names or other technical interface elements.[116] Accordingly, the court's reasoning in CMAX indicates that copyright protection extends to APIs in instances where the APIs are not dictated by industry standards, by efficiency, or by the need for a program to interact with the central host computer with which it was designed to operate.[117]

Similar to the court in CMAX, the court in Consul Tec v. Interface Systems, Inc. held that, since a great deal of original, creative effort was expended in the creation of Interface's computer program, Interface had enforceable copyrights in the "unique compilation of commands, its command line syntax, and its status message codes, all of which constitute[d] unique, creative expression, which [was] separable from the program's 'idea.'"[118] Whereas the Consul Tec court focused on the user interface of the program, it noted that, to the extent computer programs "manifest original expression, the copyright protection encompasses literal as well as non-literal aspects, including a program's source and object code, flowcharts, 'structure, sequence and organization' or 'overall structure' of the program."[119] In this respect, Consul Tec supplements the CMAX court's holding--specifically, to the extent that APIs are not dictated by industry standards, by efficiency, or by the need for a program to interact with a central host computer with which it was designed to operate, copyright protection is warranted.[120]

Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc. provides additional support for the proposition that the structure, sequence and organization of a computer ***79** program may be protected under the copyright laws.[121] In Whelan, the Third Circuit reasoned that, since the purpose or function of a utilitarian work (which encompasses computer programs) is the work's idea, everything that is not necessary to that purpose or function is part of the expression of the idea.[122] As such, where there are various means of achieving a desired purpose and the particular means chosen is not necessary to the purpose, protection is afforded under the copyright laws.[123] In this respect, Whelan provides support for the proposition that, to the extent that APIs are integral to the structure and organization of a software program, the APIs may be deemed protected expression under the copyright laws.[124]

Building on the decisions in CMAX, Consul Tec, and Whelan is Engineering Dynamics, Inc. v. Structural Software, Inc.[125], a case which involves input formats employed by end-users.[126] In Engineering Dynamics, the Fifth Circuit held that the input formats of an application program designed to solve structural engineering ***80** problems constituted copyrightable subject matter.[127] Engineering Dynamics defined a specific set of input formats for use with its program, in which the user could enter the required data, including construction details and anticipated environmental and other external forces.[128] In creating its own software, Structural Software copied many of Engineering Dynamics's input formats.[129] The Fifth Circuit found the input formats to be protected by the copyright laws, and thus reversed the district court's ruling that the input formats were uncopyrightable.[130]

The court described the input formats as "quasi-textual," consisting of a "series of words and a framework of instructions that act as prompts" to the user to provide relevant data to the program so it can perform a series of sophisticated structural analyses.[131] The court stated that "generally, functional interfaces that directly teach or guide the user's independent decisions are more expressive than functional interfaces that lack these qualities."[132] Based on its finding that plaintiff's interface "imparts knowledge" by telling the user which data to collect as well as the order of collection, the court concluded that plaintiff's input formats showed enough original expression to warrant copyright protection.[133] Although the input formats ruled protectable were part of the user interface, and it was the copyright in the user manual which the court held was infringed, the input formats, when stored, become embodied in file formats, which are part of the technical interface.[134]

In this respect, the interface ruled protectable in Engineering Dynamics includes both user interfaces and technical interfaces.[135] Since the holding that the formats are protectable relies on the user interface aspect of the formats, in that they impart knowledge to the user, it may be argued that Engineering Dynamics also protects the formats as embodied in the technical interface of the program, despite the fact that the court did not discuss or rule on the protectability of the technical interface in isolation from the user aspects.[136]

***81** Accordingly, the decisions in Control Data Systems, CMAX, Consul Tec, Whelan, and Engineering Dynamics support the contention that APIs may qualify as protected expression under the copyright laws.[137] Specifically, where there is evidence that APIs are not dictated by industry standards, efficiency or the need for a program to interact with a central host computer, copyright protection is warranted.[138] Moreover, a claim for copyright infringement of APIs may be upheld in circumstances where the APIs are not dictated by the need for interoperability with an underlying computer program.[139] Similarly, to the extent APIs manifest original expression or are integral to the structure and organization of a software program, a claim for copyright infringement is defensible.[140]

As it is evident from a review of the relevant case law, courts have adopted various approaches when faced with the question of copyright protection for APIs.[141] Nevertheless, assuming a company were successful in arguing that its APIs qualify as protected expression under the copyright laws, a competitor may be able to successfully offer a defense that copying of the APIs constitutes fair use.[142] The fair use defense will now be discussed.

IV. Fair Use and the Creation of Intermediate Copies of APIs

For purposes of this section of the essay, assume that Company B created intermediate copies of Company A's APIs in the process of creating its own software through decompilation of Company A's software. Under the Copyright Act, disassembly of copyrighted object code may be considered fair use of the copyrighted work if such disassembly provides the only means of access to those elements of the work that are not protected by copyright and the copier has a legitimate reason ***82** for seeking such access.[143] However, whereas the copyright laws may permit Company B to reverse engineer Company A's software, strict guidelines govern the method of reverse engineering that Company B may employ and the extent of information that Company B is permitted to copy.[144]

Copyrighted software programs typically contain both copyrighted expression and unprotected, functional elements.[145] Software engineers who seek to design a product that is compatible with a copyrighted product must frequently reverse engineer the copyrighted product in order to gain access to the functional elements of the copyrighted work.[146] Accordingly, courts have consistently held that, where reverse engineering is necessary to examine the unprotected ideas and functional concepts of a copyrighted work, and intermediate copies of the work are created during the reverse engineering process, Section 107 of the Copyright Act may serve to permit the copying as fair use.[147]

In determining whether a challenged use of copyrighted material is fair, courts have emphasized the public policy underlying the Copyright Act.[148] Whereas the immediate effect of the copyright laws is to secure a fair return for an author's creative labor, the ultimate aim is to stimulate artistic creativity for the general public good.[149] The court in Sega Enterprises, Ltd. v. Accolade, Inc. outlined ***83** the factors to consider when analyzing a defense of fair use.[150] These factors include:
(1) The purpose and character of the use, including whether such use is of commercial nature or is for nonprofit educational purposes;

(2) The nature of the copyrighted work;

(3) The amount and substantiality of the portion used in relation to the copyrighted work as a whole; and

(4) The effect of the use upon the potential market for or value of the copyrighted work.[151] As the Sega court notes, "The statutory factors are not exclusive."[152] Rather, the doctrine of fair use is "an equitable rule of reason."[153] Each statutory factor will now be discussed in detail.

A. Purpose and Character of Use

With respect to the first statutory factor, the court in Sega Enterprises, Ltd. v. Accolade, Inc. noted that the fact that copying is committed for a commercial purpose weighs against a finding of fair use.[154] However, such a commercial purpose does not in itself create a presumption of unfairness, but rather is a single factor that weighs in that direction.[155] For instance, courts are free to consider the public benefit resulting from a particular use, notwithstanding the fact that the alleged infringer may gain commercially.[156] As was the case in Sega, Accolade's identification of the functional requirements for compatibility, which it discovered through disassembly of Sega's copyrighted object code, led to an increase in the number of independently designed video game programs offered for use with the Sega game console.[157]

Similarly, in Connectix, Connectix's commercial use of Sony's copyrighted material--whereby Connectix reverse-engineered Sony's software for its PlayStation game console to produce a product that would be compatible with games designed for the PlayStation--was an intermediate one, and thus was only deemed ***84** "indirect or derivative" use.[158] As the Ninth Circuit noted, "it is precisely this growth in creative expression, based on the dissemination of other creative works and the unprotected ideas contained in those works, that the Copyright Act was intended to promote."[159]

The court in Connectix approached the first statutory factor by questioning whether Connectix's use "merely supercedes the objects of the original creation, or instead adds something new, with a further purpose or different character, altering the first with new expression, meaning, or message."[160] In other words, the court sought to determine whether, and to what extent, the new work is transformative.[161] Since Connectix's work created a new platform (specifically, a personal computer running its software) on which customers could play games designed for Sony's PlayStation, the court found Connectix's work to be "modestly transformative."[162] That is, Connectix's innovation afforded the opportunity for games to be played in a new environment.[163] Further, notwithstanding the similarity of uses and functions between the Sony PlayStation and Connectix's Virtual Game Station, Connectix's work resulted in an entirely new product.[164]

Accordingly, the first statutory factor requires a careful examination of the nature and function of the copied work and a determination of the relative value of the new product created in terms of benefit to the public.[165] Thus, with respect to APIs, courts must clearly identify the particular functionalities of the copied APIs and their relative role in the overall computer program, while balancing the public benefit of the newly created product.

B. Nature of Copyrighted Work

The second statutory factor reflects the fact that not all copyrighted works are entitled to the same level of protection.[166] The protection established by the Copyright Act does not extend to the ideas underlying a work or to the functional or factual aspects of the work.[167] As the Sega court notes, "To the extent that a work is ***85** functional or factual, it may be copied, as may those expressive elements of the work that 'must necessarily be used as incident to' expression of the underlying ideas, functional concepts, or facts."[168]

As a result of the nature of computer programs, they present unique problems for the application of the idea/expression dichotomy.[169] To the extent that there may exist a number of methods by which a particular task or demand may be fulfilled, a software developer's choice of program structure and design may be highly creative, and thus protected.[170] However, since computer programs are utilitarian articles, they often "contain many logical, structural and visual display elements that are often dictated by the function to be performed, by considerations of efficiency, or by external factors such as compatibility requirements and industry demands."[171]

In this respect, the hybrid nature of computer programs prevents a universal standard for distinguishing protected expression from unprotected ideas. Under Third Circuit law, for instance, "the idea or function of a computer program is the idea of the program as a whole, and everything that is not necessary to that purpose or function [is] part of the expression of that idea."[172]

As the court in Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc. highlighted, “among the more significant costs in computer programming are those attributable to developing the structure and logic of the program.”[173] Accordingly, the court held that copyright protection extends beyond the literal computer code, thus providing “the proper incentive for programmers by protecting their most valuable efforts, while not giving them a stranglehold over the development of new computer devices that accomplish the same end.”[174]

Other courts, such as the Ninth Circuit, apply a somewhat different test.[175] In particular, these courts break down a computer program into its component subroutines, and individually identify the idea and functional elements of each subroutine.[176] The remaining elements, if any, are deemed expression, and are afforded ***86** copyright protection.[177] With respect to computer programs, an analysis of the second statutory factor invariably raises the question of whether the methods used to reverse engineer the copyrighted work were necessary to gain access to the unprotected functional elements within the program.[178]

In Dun & Bradstreet Software Services, Inc. v. Grace Consulting, Inc, the Third Circuit set aside a jury verdict and granted judgment as a matter of law on Dun & Bradstreet’s claim of copyright infringement, where Grace created a competitive software package after copying and modifying Dun & Bradstreet’s program.[179] In its opinion, the court reasoned that, since Grace’s software would not work without the copied portion of Dun & Bradstreet’s program, the information Grace copied was highly critical and thus a reasonable basis for a copyright infringement claim.[180] As the court noted, the Copyright Act grants a copyright owner the exclusive right to create derivative works that are based upon the work that is copyright protected.[181] Since a derivative work is defined as “a new created work based on the original copyrighted work,” and Grace’s program utilized copies of Dun & Bradstreet’s copyrighted computer code, the court found Grace’s program to be an infringing derivative work.[182] The court was not persuaded by Grace’s claim that it copied the relevant portions of Dun & Bradstreet’s copyrighted software for purposes of interoperability.[183]

Thus, with respect to APIs, to the extent that APIs are purely functional elements of a computer program, the second statutory factor would support a defense of fair use.[184] On the other hand, where APIs play a critical role in the development and functionality of a computer program, the nature of the copyrighted work is such that a fair use defense is not likely to be upheld.[185]

*87 C. Amount and Substantiality of Copyrighted Work Used

The third statutory factor analyzes the “amount and substantiality of the portion copied in relation to the copyrighted work as a whole.”[186] Courts have generally not given much weight to this factor.[187] For instance, in Connectix, the alleged infringer disassembled parts of the plaintiff’s product and copied the software of the entire product multiple times.[188] Nonetheless, the court held that such intermediate copying, when the final product does not itself contain infringing material, is of very little weight.[189] Similarly, the court in Sega stated that the fact that an entire work has been copied does not preclude a finding of fair use.[190]

In this respect, the amount and substantiality of the APIs copied is not likely to support or preclude a defense of fair use.[191] Rather, the ultimate determination is likely to rest with factors that relate to whether the portions copied were for purposes of compatibility or as a means of creating a derivative work.[192]

D. Effect on Potential Market

The fourth statutory factor bears close relationship to the “purpose and character” inquiry.[193] Specifically, it serves to accommodate the distinction between copying of works for the purpose of furthering creative expression and exploiting the fruits of another’s creative efforts.[194] Under the fourth factor, courts must inquire whether, if the challenged use became widespread, it would adversely affect the potential market for the copyrighted work, by diminishing potential sales, interfering with marketability, or usurping the market.[195] If the copying resulted in the latter effect, all other considerations may be irrelevant.[196] However, the same consequences do not attach to a use which simply enables the copier to enter the market for works of the same type as the copied work.[197]

***88** For example, the court in Sega held that Accolade did not attempt to “scoop” Sega’s release of any particular game, but rather sought to become a legitimate competitor in the field of Sega-compatible video games.[198] As such, the minor economic loss that Sega may have suffered was not sufficient to bar Accolade’s fair use claim.[199] Similarly, in Connectix, the court reasoned that, "whereas a work that merely supplants or supercedes another is likely to cause a substantially adverse impact on the potential market of the original, a transformative work is less likely to do so."[200] Since the court found that Connectix's Virtual Game Station was transformative, and not merely a work that supplanted Sony's Playstation console, the impact on the potential market was not such that it precluded a finding of fair use.[201]

On the other hand, where an alleged infringer's work is neither creative nor transformative, and does not provide the marketplace with a new creative work, a fair use defense will fail.[202] In Triad Systems Corp. v. Southeastern Express Co., the Ninth Circuit found that the defendant did not make minimal use of plaintiff's software solely to achieve compatibility with plaintiff's computers.[203] Rather, the court held, the defendant copied plaintiff's software and invented nothing new of its own.[204] In this respect, it "undoubtedly diminished the value of [plaintiff's] copyright."[205] As the court indicated, if independent service organizations like the defendant freely used copyrighted software on a widespread basis to compete with the plaintiff, "this would likely cause a significant adverse impact on [plaintiff's] licensing and service revenues and lower returns on its copyrighted software investment."[206] Thus, there is no public benefit that would justify a fair use defense in such instances.[207]

Accordingly, the fair use defense does not justify extensive efforts to profit from replicating protected expression.[208] Further, intermediate copying does not ***89** extend to commercial exploitation of protected expression.[209] As such, the fair use reproductions of a computer program must not exceed what is necessary to understand the unprotected elements of the copyrighted work.[210] As the Atari court explains, "This limited exception is not an invitation to misappropriate protectable expression."[211] The reproduction of protectable expression, pursuant to the fair use exception, "must be strictly necessary to ascertain the bounds of protected information within the work."[212]

Conclusion

As a review of the relevant case law indicates, the extent of copyright protection afforded to APIs varies considerably depending on the specifics of the underlying computer program and the nature and function of the APIs with respect to that program.[213] To the extent APIs are not dictated by industry standards, efficiency or the need for a program to interact with a central host computer, they are likely to be afforded protection under the copyright laws.[214] Similarly, copyright protection for APIs is warranted in circumstances where the APIs manifest original expression is not integral to the structure and organization of a software program.[215]

On the other hand, the copyright laws do not protect instances where the structure of a program's APIs is dictated by external factors or where the structure of the APIs merges with the underlying function of the program itself.[216] Likewise, to the extent that copying of APIs is necessary for purposes of compatibility, a claim for copyright infringement is not likely to be upheld.[217] In situations where copying of APIs may be the basis for a claim of copyright infringement, the fair use doctrine may be utilized as a defense.[218] For these cases, a thorough examination of the nature of the APIs, in relation to the underlying computer program, must be the ***90** focus of the analysis of the statutory factors which form the basis of the fair use defense.[219]

Footnotes

a1 Member of the Bar of New York and the Bar of New Jersey; J.D., University of Pennsylvania Law School (2000); M.BE., University of Pennsylvania Center for Bioethics (2000); B.A. in Philosophy, The College of New Jersey (1997). The views expressed herein are strictly those of the author.

1 See, e.g., Daniel J. Gifford, Antitrust's Troubled Relations with Intellectual Property, 87 Minn. L. Rev. 1695 (2003); Raymond T. Nimmer & Patricia Ann Krauthaus, Software Copyright: Sliding Scales and Abstracted Expression, 32 Hous. L. Rev. 317, 318 (1995); Kenneth W. Dam, Some Ecomonic Considerations in the Intellectual Property Protection of Software, 24 J. Legal Stud. 321 (1995). Compare with the framework regarding application program interfaces ("APIs") as promulgated by the European Union under the Framework Directive, which promotes the idea of technical standardization with a view to achieving harmonization and interoperability of electronic communications networks, electronic communications services, and associated facilities and services. See Tarlach McGonagle, The Potential for Practice of an Intangible Idea, 13 Media L. & Pol'y 28, 46-47 (2003) (citing Directive 2002/19/EC of the European Parliament and of the Council of 7 March 2002); Hernan Galperin & François Bar, The Regulation of Interactive Television in the United States and the European Union, 55 Fed. Comm. L.J. 61, 76

(2002); Kurt Wimmer & Keith Lieberthal, What Yankee Lawyers Need to Know About European Telecommunications, 20 Comm. Law. 17, 19 (2002). In the interests of interoperability, Article 18 of the Framework Directive requires Member States to encourage the use of, and compliance with, open APIs. See McGonagle, supra, at 47. Indeed, it has been proposed that similar legislation should be adopted in the United States. See, e.g., Mark A. Lemley, Standardizing Government Standard-Setting Policy for Electronic Commerce, 14 Berkeley Tech. L.J. 745, 757 (1999).

2 See, e.g., Gifford, supra note 1, at 1695. Laws surrounding the extent of copyright protection for APIs vary considerably between and among the federal district and circuit courts. Id. See also Deborah F. Buckman, Annotation, Copyright Protection of Computer Programs, 180 A.L.R. Fed. 1, § 2(a) (on the existence of judicial uncertainty regarding copyright protection for computer software, particularly APIs) (2004). Similarly, courts have offered differing perspectives on the nature and purpose of copyright protection for APIs. See Gifford, supra note 1, at 1695; Lemley, supra note 1, at 751 n. 26. Since determining the extent of copyright protection for APIs significantly influences other areas of the law, such as antitrust, there is a need to clarify the level of copyright protection afforded to APIs. See, e.g., Gifford, supra note 1, at 1695; Aaron S. Edlin, Stopping Above-Cost Predatory Pricing, 111 Yale L.J. 941, 989 (2002); David A. Balto, Standard Setting in the 21st Century Network Economy, 6 Computer & Internet Law., June 2001, at 5, 16; Summary of D.C. Circuit Opinion in United States v. Microsoft, 18 No. 9 Computer & Internet Law. 26 (2001); Timothy J. Brennan, Do Easy Cases Make Bad Law? Antitrust Innovations or Missed Opportunities in United States v. Microsoft, 69 Geo. Wash. L. Rev. 1042, 1062-63 (2001).

3 United States v. Microsoft Corp., 253 F.3d 34 (D.C. Cir. 2001), cert. denied, 534 U.S. 952 (2001); United States v. Microsoft Corp., 97 F. Supp. 2d 59 (D.D.C. 2000); United States v. Microsoft Corp., 87 F. Supp. 2d 30 (D.D.C. 2000); United States v. Microsoft Corp., 84 F. Supp. 2d 9 (D.D.C. 1999); United States v. Microsoft Corp., 65 F. Supp. 2d 1 (D.D.C. 1999). See also Gifford, supra note 1, at 1715-16. It has been argued that, with respect to United States v. Microsoft Corp., 253 F.3d 34 (D.C. Cir. 2001), the relevant monopoly for antitrust purposes may not have been the Windows operating system at all. Gifford, supra note 1, at 1715-16. Rather, the relevant monopoly might be better stated as the operating system's APIs. See id. It is these interfaces that software developers must access and that produce the network effects that make them the industry interface standard. Id. Indeed, with respect to Microsoft, 253 F.3d 34, Microsoft executives were genuinely concerned that Java or Netscape might eventually undermine Windows's dominance of the operating system market because Java and Netscape utilized APIs for their respective software packages that would have allowed the software to run on several different operating systems at the same time. See Max Schanzenbach, Network Effects and Antitrust Law: Predation, Affirmative Defenses and the Case of U.S. v. Microsoft, 2002 Stan. Tech. L. Rev. 4 (2002). This, in turn, would have posed a significant threat to Microsoft's position in the market and would have decreased the value of the Windows operating system. Id. As such, Microsoft went to great lengths to diminish the relative value of its competition, including an active design in its own APIs which significantly limited the ability of its competitors to create software interoperable with the Windows platform. Id. See also Brendan Dowd, Andrew Frackman & Matthew Merrick, Current Developments in Sherman Act Section 2 Exclusionary Conduct Cases, 2003 Colum. Bus. L. Rev. 526, 541 (2003); Benjamin Klein, Exclusive Dealing as Competition for Distribution "On the Merits," 12 Geo. Mason L. Rev. 119, 137 n. 52 (2003) (quoting United States v. Microsoft Corp., 253 F.3d at 71 ("Microsoft's only explanation for its exclusive dealing is that it wants to keep developers focused upon its APIs--which is to say, it wants to preserve its power in the operating system market.")); Summary of D.C. Circuit Opinion in United States v. Microsoft, supra note 2, at 26 (highlighting the role of APIs in the Circuit Court's opinion).

4 See, e.g., Robert L. Bocchino, Jr., Computers, Copyrights and Functionality: The First Circuit's Decision in Lotus Development Corp. v. Borland International, Inc., 9 Harv. J.L. & Tech. 467, 467 (1996) (discussing the limits of copyright protection for computer software under the copyright laws and the lack of coherent justifications with respect to the nature and extent of copyright protection for functional elements of a program); Nimmer, supra note 1, at 318.

5 Harry Newton, Newton's Telecom Dictionary 52 (17th ed. 2001). See also Brennan, supra note 2, at 1056 n. 77. An operating system is defined as a "software program that runs the computer by assigning memory and allotting tasks." Schanzenbach, supra note 3, at 16. Operating systems support other software programs through use of the system's APIs. Id. Thus, in creating new software, programmers can call upon existing APIs, rather than writing new routines to perform basic tasks. Id. This significantly reduces the costs of software development. Id. Without functionally identical APIs, software programs will not run on multiple operating systems. Id. For instance, software designed for Microsoft Windows will not interoperate with a competing operating system unless the APIs are functionally identical. Id. See also Application Programming Interface, http://en.wikipedia.org/wiki/API (last visited Feb. 12, 2006) ("APIs, like most interfaces, are abstract. Software that may be accessed via a particular API is said to implement that API.").

6 In order to employ the functionality of a computer language or an operating system, applications must identify specific instructions to the operating system in accordance with the system's rules or syntax. See Mitchell Zimmerman, Baystate Holding: Technical

Interfaces Not Copyrightable, 2 (1997), http://www.fenwick.com/docstore/Publications/IP/IP_Articles/Baystate_Holding. For instance, "to draw a line on a screen, it is necessary to state in a specified order, using specified terms, within a specified range, the starting and ending points, color, thickness and other characteristics of the line." Id. Such a parameter structure would be defined by an API. Id.

7 Operating system calls, which are a type of function call, allow application programs to invoke certain functionality of the underlying operating system by specifying certain words or commands in the source code of the program. Id. at 1-2. These commands are defined by the operating system's APIs. See also Keith Stephens & John P. Sumner, Software Objects: A New Trend in Programming and Software Patents, 12 Santa Clara Computer & High Tech. L.J. 1, 6 n. 14, 8 n. 17 (1996).

8 Such information includes input and output formats. Zimmerman, supra note 6, at 2. For instance, software that permits the user to input data (such as statistical information for regression analysis) often produces data as an output for use by other programs (such as figures for a power point presentation). Id. As Zimmerman notes:
Input and output formats specify the ordering, structure and type of data that must be input or that can be output by the program. Input data may be input manually, passed in from another program or stored in a file. When stored in a file, input or output data is stored in a specified file structure, and for data to be passed from another program, usually either that program must employ the same file employing the same formats or structure or must be translated into a readable structure.
Id. In this respect, input and output formats are properly characterized as APIs. Id.

9 See Alan J. Meese, Intrabrand Restraints and the Theory of the Firm, 83 N.C. L. Rev. 5, 82 n. 388 (2004); Newton, supra note 5 at 52 (Definition of "API"). See also API, http://computing-dictionary.thefreedictionary.com/API (last visited Feb. 12, 2006); Application Programming Interface, http://en.wikipedia.org/wiki/API (last visited Feb. 12, 2006) (explaining that a computer program often uses APIs of the operating system to allocate memory and access files.)

10 See Ronald A. Cass & Keith N. Hylton, Antitrust Intent, 74 S. Cal. L. Rev. 657, 723-24 (2001). The size and complexity of a particular API can influence the number of programs that are written to interact with it. See, e.g., Alan J. Meese, Don't Disintegrate Microsoft (Yet), 9 Geo. Mason L. Rev. 761, 773-74 (2001). For instance, as of 1998, the APIs of Netscape's Navigator and Sun Microsystem's JavaClass Libraries, combined, consisted of fewer than 1000 function calls, compared to approximately 10,000 function calls for the APIs of Microsoft Windows 98. Id. at 773 n. 69. As Judge Jackson indicates, "the large number of application programs written to the APIs exposed by Windows hindered competition from rival operating systems, such as Apple's Mac OS, which could not run these applications." Jonathan Band, Paragraph 52: A Window into Judge Jackson's Finding of Fact, 17 Computer Law. 3, March 2000 at 3, 3 (citing United States v. Microsoft Corp., 65 F. Supp. 2d 1 (D.D.C. 1999)). As Band explains, "Judge Jackson's 207-page Findings of Fact ... recite in great detail Microsoft's efforts to preserve the API barrier to entry by preventing the broad adoption of 'middleware,' such as Netscape's Navigator, which would run on top of Windows and expose a different and attractive set of APIs to independent software developers." Id.

11 Newton, supra note 5 at 52. (Definition of "API"). See also Galperin, supra note 1, at 75 (APIs include the software layer between an operating system and the different applications that run on the system); Meese, supra note 9, at 82 n. 388 (explaining that APIs are software code which authors of applications utilize when creating new applications). By utilizing APIs that are interoperable with multiple software programs, software developers are able to build upon, and adapt to, each other's work. See Stephens, supra note 7, at 8. See also Application Programming Interface, http://en.wikipedia.org/wiki/API (last visited Feb. 12, 2006) (explaining that software development kits often include APIs). Some developers elect to keep the APIs exactly the same, while others select to make extensive modifications. See Stephens, supra note 7, at 8. As Stephens notes, "this framework system architecture provides flexibility and increased extensibility, while increasing functionality." Id.

12 See Ryan Roemer, Trusted Computing, Digital Rights Management, and the Fight for Copyright Control on Your Computer, 2003 UCLA J. L. & Tech. 8 n. 103 (2003); API, http://www.webopedia.com/TERM/A/API.html (last visited Feb. 12, 2006). In connection with the Microsoft case, a May 1995 memo from Bill Gates emerged which highlighted the relative importance of APIs with respect to Netscape's Navigator program, launched in December 1994. See United States v. Microsoft Corp., 84 F. Supp. 2d 9, 72 (D.D.C. 1999). In the memo, Mr. Gates warned his colleagues at Microsoft that Netscape was "pursuing a multi-platform strategy where they move the key API into the client to commoditize the underlying Operating System." Id. See Thomas M. Lenard, Creating Competition in the Market for Operating Systems: Alternative Structural Remedies in the Microsoft Case, 9 Geo. Mason L. Rev. 803, 811 (2001). As Lenard explains, "[a]ny development that 'commoditized the Operating System' would neutralize the power of the Windows monopoly both as a leading source of Microsoft's extraordinary profits and as a source of leverage into other markets." Id.

13 Newton, supra note 5 at 52 (Definition of "API"). The success of a network is based, in part on interoperability, which is defined as "the capacity of products of one vendor to communicate or interface with the products of competing suppliers of complementary products." Balto, supra note 2, at 6. Whereas no network is an island, networks must depend on alliances with producers of complementary products for maximum performance. Id. In this respect, interoperability is a core function of most information technology products. Id. For example, network products (such as modems and cellular phones) are heavily dependent on interoperability standards. Id.

14 Definition of "application program interface," http:// searchexchange.techtarget.com/sDefinition/0,,sid43_gci213778,00.html (last visited Jan. 27, 2006). Compare with user interfaces, such as a graphical user interface or a command interface, which are interfaces to an operating system or a program. Id. Indeed, some software developers purposely reveal their APIs in an effort to encourage other companies to create new software that is compatible with the original, while others refrain from revealing their APIs in order to stifle competition. See Meese, supra note 10, at 761. For example, in 1994, Netscape began including with its Navigator software the APIs which were required to write applications that were compatible with Navigator. Plaintiff's Memorandum in Support of Proposed Final Judgment at 16, United States v. Microsoft Corp, 97 F. Supp. 2d 59 (D.D.C. 2000) (Nos. CIV. A. 98-1232 (TPJ), CIV. A. 98-1233 (TPJ)). See also United States v. Microsoft Corp., 84 F. Supp. 2d 9 (D.D.C. 1999). At the same time, Sun Microsystems was developing Java, a programming language that could empower Independent Software Vendors (ISVs) to write one software program that would be compatible with any operating system. United States v. Microsoft Corp., 84 F. Supp. 2d at 68-77. Merely writing a program in the Java language, however, would not ensure that an application ran on differing operating systems. Id. Rather, applications written in Java would only run on operating systems that contained so-called Java Class Libraries--APIs on which ISVs writing in Java could rely, as well as a so-called Java Virtual Machine, which helped translate Java software code into instructions comprehensible to the underlying operating system. Id. Microsoft apparently refused to distribute the Java Class Libraries and the Java Virtual Machine along with Windows. See id. In 1995, however, Netscape agreed to include Java Class Libraries, as well as the Java Virtual Machine, along with its Navigator browser. Id. Sun, however, was far from its goal of including enough APIs in its Java Class Libraries to allow for a true "write once, run anywhere" environment. Id.

15 See Andrew Johnson-Laird, Software Reverse Engineering in the Real World, 19 U. Dayton L. Rev. 843, 843-45 (1995); E. J. Byrne, A Conceptual Foundation for Software Re-Engineering 226-235 (1992). See also Buckman, supra note 2, at § 2(a); Definition of "decompile," http:// searchvb.techtarget.com/sDefinition/0,,sid8_gci804135,00.html (last visited Jan, 27, 2006); Cristina Cifuentes & K. John Gough, Decompilation of Binary Programs, 25 Software Prac. & Experience 811, 811-812 (1995).

16 Johnson-Laird, supra note 15, at 843-45. See Band, supra note 10, at 3. See also infra, Section V (discussion of fair use defense). However, depending on the size and complexity of the APIs for a particular operating system, reverse engineering may not be a reasonable option. Band, supra note 10, at 3. In his Findings of Fact of the Microsoft case, Judge Jackson provides a concise explanation for this scenario:

Theoretically, the developer of a non-Microsoft, Intel-compatible PC operating system could circumvent the applications barrier to entry by cloning the APIs exposed by the 32-bit versions of Windows (Windows 9x and Windows NT). Applications written for Windows would then also run on the rival operating system, and consumers could use the rival system confident in that knowledge. Translating this theory into practice is virtually impossible, however. First of all, cloning the thousands of APIs already exposed by Windows would be an enormously expensive undertaking. More daunting is the fact that Microsoft continually adds APIs to Windows through updates and new versions. By the time a rival finished cloning the APIs currently in existence, Windows would have exposed a multitude of new ones. Since the rival would never catch up, it would never be able to assure consumers that its operating system would run all of the applications written for Windows. IBM discovered this to its dismay in the mid-1990s when it failed, despite a massive investment, to clone a sufficiently large part of the 32-bit Windows APIs. In short, attempting to clone the 32-bit Windows APIs is such an expensive, uncertain undertaking that it fails to present a practical option for a would-be competitor to Windows.

United States v. Microsoft Corp., 65 F. Supp. 2d 1, 15 (D.D.C. 1999). See Band, supra note 10, at 3. Indeed, it is intriguing to consider how different the computer industry would be today if IBM had succeeded in its effort to reverse-engineer Windows. See Band, supra note 10, at 5. For a historical precedent, one can consider Phoenix Technologies's reverse engineering of the IBM PC's basic input/output system (BIOS) in the 1980s. See id. With this information, Phoenix developed a compatible BIOS, which it made available to other hardware manufacturers. See id. This contributed significantly to the IBM-compatible PC industry and the ultimate rise of companies such as Dell and Gateway. See id.

17 Cifuentes, supra note 15, at 811-12.

18 Id.

19 Id.

20 Id. Decompilation is not always successful for a number of reasons. It is not possible to decompile all programs, and data and code are difficult to separate, because both are represented similarly in most current computer systems. Id. The meaningful names that programmers give variables and functions (to make them more easily identifiable) are not usually stored in an executable file, so they are not usually recovered in decompiling. Id. Furthermore, decompilation is sometimes used unethically to reproduce source code for reuse or adaptation without permission of the copyright holder. Id. Programs can be designed to be resistant to decompilation through protective means such as obfuscation. Id.

21 Id. at 814; Dan L. Burk & Mark A. Lemley, Policy Levers in Patent Law, 89 Va. L. Rev. 1575, 1692-93 (2003). See also Buckman, supra note 2, at § 2(a) ("The nature of computer programs in particular is such that intermediate copying is sometimes required to understand the ideas and processes therein.").

22 Burk, supra note 21, at 1692-93. To establish a claim of copyright infringement, a plaintiff must demonstrate: (1) ownership of a valid copyright; and (2) unauthorized "copying" of original elements of the plaintiff's work by the defendant. Whelan Assocs, Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986). Because it is only in rare circumstances that there will be direct evidence of copying, copying may be proven indirectly by establishing that a defendant had access to the copyrighted work and that there is a substantial similarity between the copyrighted work and the defendant's work. See id. To prove ownership of a valid copyright, registration with the United States Copyright Office serves as prima facie evidence of that ownership. Although original computer programs and other original works of authorship created after 1977 are automatically "copyrighted" at the moment they are created, without regard to whether they are ever registered, registration is a prerequisite for enforcement of a copyright infringement claim. 17 U.S.C. § 412 (2000). Regarding the second element of an infringement claim involving computer software, "copying" has been called the "shorthand for the infringing of any of the copyright owner's five exclusive rights." S.O.S., Inc. v. Payday, Inc., 886 F.2d 1081, 1085 n.3 (9th Cir. 1989). These statutorily dictated exclusive rights are, for literary works, to reproduce the copyrighted work in copies, to prepare derivative works based thereon, to distribute copies, and to perform and display the work publicly. 17 U.S.C. § 106 (2000). Violation of any of these rights will generally constitute copyright infringement, unless an exception or defense to an infringement claim exists. See, e.g., Band, supra note 10, at 4. For example, copying APIs incidental to decompilation may be deemed fair use so long as decompilation is the only way to uncover the information and the decompilation was performed for a legitimate purpose. Id. (citing Sega v. Accolade, 977 F.2d 1510 (9th Cir. 1992)). Fair use is discussed infra, Section V.

23 U.S. Const. art. I, § 8, cl. 8. The copyright laws have been enacted to promote a "delicate equilibrium." Buckman, supra note 2, at § 2(a). As Buckman explains, "[o]n the one hand, the laws grant protection to authors as a stimulus for artistic creativity for the public good, while, on the other hand, appropriately limit the extent of that protection so as to avoid the effects of monopolistic stagnation." Id. See also Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340 (1991).

24 Encouraging the production of original works is the fundamental purpose of the Copyright Act. See Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc., 499 U.S. 340 (1991). This is accomplished through protection of expressive elements of works, while leaving ideas, facts, and functional concepts in the public domain for others to build upon. Id. See also Buckman, supra note 2, at § 2(a).

25 At the time of the enactment of the Copyright Act, Congress was hesitant about expanding the Act to incorporate computers, pending further study by the Commission On New Technological Uses of Copyright Works (CONTU). Buckman, supra note 2, at § 2(a). Thus, in 1980, subsequent to the final CONTU report, which was published in 1978, Congress adopted an amendment to the Copyright Act which expressly included computer programs as proper subjects of copyright. Id. Specifically, the Amendment added the definition of "computer program" to the definitional section of copyrightable subject matter and replaced section 117 of the 1976 Act with a new section regarding the exclusive rights given to "authors" in copies of computer programs. Id. As such, the Copyright Act currently defines a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101 (2000). Computer programs are afforded copyright protection as "literary works," within the definition, 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, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied." Id. See, e.g., AccuSoft Corp. v. Mattel, Inc., 117 F. Supp. 2d 99 (D. Mass. 2000).

26 Buckman, supra note 2, at § 2(a). There exist a number of doctrines, in addition to statutory provisions within the Copyright Act, that address the idea/expression distinction. Id. The fair use doctrine, which is codified in section 107 of the Copyright Act, is one example. Id. The fair use doctrine allows for reproductions of copyrighted material where the copying is necessary to uncover basic ideas of a copyrighted work. Id. See also Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 599 (9th Cir. 2000). Fair use is discussed infra, Section V.

27 Connectix, 203 F.3d at 599. See also J. Dianne Brinson, Copyrighted Software: Separating the Protected Expression from Unprotected Ideas, A Starting Point, 29 B.C. L. Rev. 803 (1988).

28 17 U.S.C. § 102(b) (2000); Apple Computer, Inc. v. Microsoft Corp., 799 F. Supp. 1006, 1023 (N.D. Cal. 1992), aff'd, 35 F.3d 1435 (9th Cir. 1996). See also Brinson, supra note 27, at 803. Under section 102(b) of the Copyright Act, copyright protection does not "extend to any ... procedure, process, system or method of operation." Rather, such protection is afforded under the patent laws. Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 839 (Fed. Cir. 1992) (quoting 17 U.S.C. § 102(b)).

29 Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1547 (11th Cir. 1996).

30 Id. See also Gifford, supra note 1, at 1716. APIs may not qualify for protection under the copyright laws if the APIs are deemed an unprotected "system of operation" under section 102(b) of the Copyright Act. See id. It is interesting to note that, in the Microsoft case, the court of appeals gave merit to Microsoft's claim that the bundling of its Internet Explorer APIs with its Windows operating system may indeed make the Windows operating system a better application platform for third-party software. United States v. Microsoft Corp., 253 F.3d 34, 90 (D.C. Cir. 2001); James B. Speta, Maintaining Competition in Information Platforms: Vertical Restrictions in Emerging Telecommunications Markets, 1 J. Telecomm. & High Tech. L. 185, 192 (2002). The court went on to indicate that, a per se rule against the combination of previously separate products would "chill innovation to the detriment of consumers by preventing firms from integrating into their products new functionality previously provided by standalone products." United States v. Microsoft Corp., 253 F.3d at 89-93. See also Speta, supra, at 192-193. Nevertheless, the court of appeals did affirm much of the government's monopoly maintenance theory, and some of the specific practices challenged were themselves tying requirements centering on Microsoft's APIs. United States v. Microsoft Corp., 253 F.3d at 60-64. See also Speta, supra, at 193. Furthermore, the functional aspects of a computer program do not preclude copyrightability for every part and aspect of that program. See Buckman, supra note 2, at § 2(a). A trial court's initial task is separating protectable expression of ideas in a computer program from unprotectable ideas, facts, processes, and methods of operation. Id. However, "despite its clear mandate to make this important distinction, the Copyright Act contains no explicit standards for separating a work's expression from its underlying idea." Id.

31 See Bateman, 79 F.3d at 1539-40; Baystate Techs., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1094 (D. Mass. 1996); Mitel Inc. v. Iqtel, Inc., 896 F. Supp. 1050 (D. Colo. 1995).

32 Bateman, 79 F.3d at 1546-47.

33 See Baystate, 946 F. Supp. at 1088; Mitel, 896 F. Supp. at 1057.

34 Bateman, 79 F.3d at 1539-40.

35 Id.

36 Id. at 1537.

37 Id. at 1536.

38 Id. at 1546-47. The court further indicated that, even if the elements were found to be protected expression, copying such elements

constitutes fair use. Id. The defense of fair use is further discussed infra, Section V.

39 Bateman, 79 F.3d at 1546-47.

40 Id.; Mitel Inc. v. Iqtel, Inc., 896 F. Supp. 1050 (D. Colo. 1995). Since Mitel does not directly involve computers or computer programs, however, there can be some argument about whether it is applicable to APIs. Mitel, 896 F. Supp. at 1050. Further, it may be argued that Mitel involves a user interface and user compatibility expectations, rather than a technical interface and technical compatibility requirements. Id. at 1052-53. Notwithstanding the imperfect fit, however, the case supports an argument that technical interfaces are not protectable expression under the copyright laws. Id. at 1057.

41 Mitel, 896 F. Supp. at 1050.

42 Id. at 1052-53.

43 Id.

44 Id.

45 Id. at 1053.

46 Id. at 1057.

47 Id. at 1054.

48 Id. at 1056. The doctrine of scenes a faire denies copyright protection to those elements of a program that are dictated by external factors. See Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986); Baystate Techs., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1088 (D. Mass. 1996). For computer programs, such external factors include, among other things, compatibility requirements and industry-wide programming practices. Baystate, 946 F. Supp. at 1088. Essentially, scenes a faire prohibits an author from receiving a de facto monopoly through the copyright laws. Id.

49 Mitel, 896 F. Supp. at 1055.

50 With respect to our paradigm, Mitel adds significant weight to the contention of Company B that a claim of copyright infringement of Company A's APIs should not be upheld. In particular, the Mitel court rejected the contention that command codes represented protectable expression because there were alternative expressions that were available for the idea of function-invoking codes, and the creator of the original work therefore made professional choices. See id. at 1056. In addition to focusing on the functionality of the command codes, the court indicated that the scenes a faire doctrine would also preclude a claim for copyright infringement. Id.

51 Id. at 1055.

52 Id.

53 Id.

54 Compare Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1546-47 (11th Cir. 1996), with Mitel, 896 F. Supp. at 1055.

55 Baystate Techs., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1094 (D. Mass. 1996). See also Bateman, 79 F.3d at 1546-47; Mitel, 896 F. Supp. at 1055.

56 Baystate, 946 F. Supp. at 1094. Baystate is important for two reasons. See Zimmerman, supra note 6, at 1. First, the protectability of technical interface elements required to create compatible works was clearly posed and resolved in the case. Id. Second, the Baystate opinion is more completely reasoned than any of the other decisions that have treated technical interface compatibility issues. Id. Since the parties recently settled the case on appeal, the district court opinion stands as the decision most sharply addressing the protectability of technical interfaces. Id.

57 Baystate, 946 F. Supp. at 1082.

58 Id.

59 Id. at 1084-85.

60 Id. at 1087-88.

61 Id. at 1085.

62 Id. at 1086.

63 Id.

64 Id.

65 Id. at 1087-88. For a discussion of the scenes a faire doctrine, see supra note 48 and accompanying text.

66 Baystate, 946 F. Supp. at 1087-88.

67 Id.

68 Id.

69 Id.

70 Id.

71 Id.

72 Baystate, 946 F. Supp. at 1087-88.

73 Id.

74 Id. at 1089-90.

75 Id. at 1087.

76 Id. at 1089.

77 Id. at 1090.

78 Baystate, 946 F. Supp. at 1090.

79 Id. With respect to Company A's claim of copyright infringement, Baystate stands for the proposition that technical specifications of a computer program, where the structure merges with the idea or function of the program itself, are not protected by the copyright laws. Whereas Baystate does not advocate a new theory of copyright protection for APIs, it does provide an analysis which implies that interface specifications should not be afforded copyright protection. See id. at 1088. That is, as a result of the nature and function of APIs, the doctrines of merger and scenes a faire are such that they typically preclude copyright protection for the APIs of a computer program. See id.

80 Id. at 1087-90; Bateman 79 F.3d at 1546-47; Mitel 896 F. Supp. at 1057.

81 See Bateman, 79 F.3d at 1546-47.

82 See Baystate, 946 F. Supp. at 1088; Mitel, 896 F. Supp. at 1057.

83 See Baystate, 946 F. Supp. at 1088; Mitel, 896 F. Supp. at 1057.

84 See, e.g., Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1348 (5th Cir. 1994).

85 Id.

86 See Bateman, 79 F.3d at 1546-47; Baystate, 946 F. Supp. at 1087-88; Mitel, 896 F. Supp. at 1055-57.

87 See Eng'g Dynamics, 26 F.3d at 1351; Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986); Control Data Sys., Inc. v. Infoware, Inc., 903 F. Supp. 1316, 1320 (D. Minn. 1995); CMAX/Cleveland, Inc. v. UCR, Inc., 804 F. Supp. 337, 340 (M.D. Ga. 1992); Consul Tec, Inc. v. Interface Sys., Inc., 22 U.S.P.Q.2d (BNA) 1538, 1541 (E.D. Mich. 1991).

88 Control Data Sys., 903 F. Supp. at 1326.

89 Id. at 1320.

90 Id. at 1321-22, 1324.

91 Id. at 1319 n.1, 1321-22.

92 Id. at 1322.

93 Id. at 1322-24.

94 Control Data Sys., 903 F. Supp. at 1323. Under the merger doctrine, copyright protection for elements of a computer program whose expression is necessarily dictated by the underlying subject matter of the software is prohibited. See Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3rd Cir. 1983). See also Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 814-16 (1st Cir. 1995). Thus, with respect to our paradigm, if there are a limited number of ways in which Company A may configure its APIs, a court is likely to hold that any protected expression that Company A has exhibited in the APIs has merged with the ideas underlying the APIs, thus precluding copyright protection for the APIs.

95 Control Data Sys., 903 F. Supp. at 1322-24.

96 Id.

97 See id. Thus, in connection with our paradigm, Control Data Systems indicates that a determination that Company A's APIs constitute protectable expression must withstand the merger doctrine. See id. Specifically, Company A must demonstrate that its APIs are not dictated by the underlying subject matter of the software. See id. Such may be proved by providing the court with alternative means of configurations for APIs that are compatible with Company A's device. See id.

98 CMAX/Cleveland, 804 F. Supp. 337, 340 (M.D. Ga. 1992).

99 Id.

100 Id. at 343.

101 Id.

102 Id. at 344.

103 Id.

104 CMAX/Cleveland, 804 F. Supp. at 354-55.

105 Cf. id.

106 Id. at 348-49, 354-55.

107 Id. at 354-55.

108 Id.

[109] Cf. id. at 354.

[110] CMAX/Cleveland, 804 F. Supp. at 354-55.

[111] Id. at 355.

[112] Id.

[113] Id.

[114] Id. at 349 n.8.

[115] See id. at 355 (referring to testimony that explained that changing the transaction codes would mean that employees using the software would have to learn the new code).

[116] Compare CMAX/Cleveland with Mitel Inc. v. Iqtel, Inc., 896 F. Supp. 1050, 1055 (D. Colo. 1995). Using the court's reasoning in CMAX as an analogy, Company A may argue that its APIs constitute protectable expression if it can demonstrate that its APIs are not dictated by industry standards, by efficiency or by the need for the program to interact with a central host computer. Thus, in order to prevail on a claim of copyright infringement with respect to Company B's copying of Company A's APIs, Company A must demonstrate that the unique nature and organization of its APIs is such that it would qualify as protected expression under the Copyright laws. Cf. CMAX/Cleveland, 804 F. Supp. at 355. That is, where the individual application of a computer program is customized to the needs of individual purchasers, the structure of the program is deemed protected expression. See, e.g., Consul Tec, Inc. v. Interface Sys., Inc., 22 U.S.P.Q.2d (BNA) 1538, 1541 (E.D. Mich. 1991).

[117] See CMAX/Cleveland, 804 F. Supp. at 354-55.

[118] Consul Tec, Inc., 22 U.S.P.Q.2d (BNA) at 1541.

[119] Id. at 1540 (citations omitted). In this respect, if Company A can demonstrate that its APIs incorporate a unique compilation of commands whereby a great deal of creative effort was needed to create the specifications, a claim of copyright protection may be defensible.

[120] CMAX/Cleveland, 804 F. Supp. at 354-55; Consul Tec, 22 U.S.P.Q.2d at 1539-40.

[121] Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d Cir. 1986).

[122] See Whelan, 797 F.2d at 1236. The Third Circuit's decision in Whelan came under scrutiny after its publication. See, e.g., Charles Walter, Defining the Scope of Software Copyright Protection for Maximum Public Benefit, 14 Rutgers Computer & Tech. L.J. 1, 130-131 (1988); J. Scott MacKay, Broderbund Software, Inc. v. Unison World, Inc.: "Look and Feel" Copyright Protection for the Display Screens of An Application Microcomputer Program, 13 Rutgers Computer & Tech. L.J. 105, 129 n. 181 (1987). However, much of this criticism is based on a misguided analysis of the function and purpose of APIs and the specific role of APIs in relation to a particular underlying computer system. See Carl A. Sundholm, High Technology Jurisprudence: In Defense of "Look and Feel" Approaches to Copyright Protection, 8 Santa Clara Computer & High Tech. L.J. 209, 219 n.43 (1991). For instance,
MacKay argues for non-protection of utilitarian aspects of the interface under the "useful article" doctrine, which holds that the only elements of an application program's interface eligible for copyright protection are those that can be separated from, and exist independently of, the utilitarian aspects of the interface. The problem with this type of approach is that it fails to understand that,

due to the fundamental difference between literary/artistic works and computer programs/displays, most aspects of computer subject matter are useful or utilitarian. This type of approach impractically restricts copyright protection to the useless and valueless aspects of the computer subject matter. Thus the "useful article" doctrine holds little promise as "the" test for computer copyright infringement cases.
Id. at 220 n.43.

123 See Whelan, 797 F.2d at 1236. The Whelan court further states that, "among the more significant costs in computer programming are those attributable to developing the structure and logic of the program." Id. at 1237. In this respect, a rule granting copyright protection to non-literal elements of a computer program that are not required by the form or function of the program provides the "proper incentive for programmers by protecting their most valuable efforts, while not giving them a stranglehold over the development of new computer devices that accomplish the same end."

124 See id.

125 Eng'g Dynamics, Inc. v. Structural Software, Inc. 26 F.3d 1335 (5th Cir. 1994).

126 Id. at 1351.

127 Id.

128 Id. at 1339.

129 Id.

130 Id. at 1351.

131 Eng'g Dynamics, 26 F.3d at 1342.

132 Id. at 1346.

133 Id.

134 Id. at 1339-42.

135 Id.

136 Id. Thus, in our paradigm, Company A would add significant value to its claim of copyright infringement if it can demonstrate that its APIs contain functions which directly teach or guide the user's independent decisions, rather than merely providing purely functional specifications. See, e.g., id.

137 Eng'g Dynamics, 26 F.3d at 1351 (5th Cir. 1994); Whelan 797 F.2d at 1236; Control Data Sys., 903 F. Supp. at 1320; CMAX/Cleveland, 804 F. Supp. at 356; Consul Tec, 22 U.S.P.Q.2d (BNA) at 1539-40.

138 See Eng'g Dynamics, 26 F.3d at 1351; Control Data Sys., 903 F. Supp. at 1320; CMAX/Cleveland, 804 F. Supp. at 356; Consul Tec, 22 U.S.P.Q.2d (BNA) at 1539-40.

139 See Control Data Sys., 903 F. Supp. at 1323.

140 See Eng'g Dynamics, 26 F.3d at 1351; Whelan, 797 F.2d at 1236; Consul Tec, 22 U.S.P.Q.2d (BNA) at 1539-40.

141 See Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1546-47 (11th Cir. 1996); Eng'g Dynamics, 26 F.3d at 1351; Whelan, 797 F.2d at 1236; Baystate Techs., Inc. v. Bentley Sys., Inc., 946 F. Supp. 1079, 1094 (D. Mass. 1996); Control Data Sys., 903 F. Supp. at 1320; Mitel Inc. v. Iqtel, Inc., 896 F. Supp. 1050, 1050-51 (D. Colo. 1995); CMAX/Cleveland, 804 F. Supp. at 352; Consul Tec, 22 U.S.P.Q.2d (BNA) at 1539-40.

142 See Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510, 1518 (9th Cir. 1993).

143 Id. See also Sony Computer Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 604 (9th Cir. 2000). Even apart from section 102(b), several copyright fair use cases indicate that APIs are unprotectable as against software providers. Gifford, supra note 1, at 1716.

144 See Sega, 977 F.2d at 1518; Connectix, 203 F.3d at 604; Johnson-Laird, supra note 15, at 845-47.

145 See Connectix, 203 F.3d at 599 (citing Sega, 977 F.2d at 1520).

146 See Connectix, 203 F.3d at 599. Copying APIs incidental to decompilation is a fair use so long as decompilation is the only way to uncover the information and the decompilation was performed for a legitimate purpose. Band, supra note 10, at 4 (citing Sega, 977 F.2d at 1510).

147 Connectix, 203 F.3d at 599; Sega, 977 F.2d at 1520; Johnson-Laird, supra note 15, at 845-47.

148 Connectix, 203 F.3d at 603; Sega, 977 F.2d at 1520.
The "fair use doctrine" ... seeks to apply the expression/ideas dichotomy by preserving public access to the ideas and functional elements embedded in copyrighted computer software programs. This limitation on a copyright holder's exclusive rights allows courts to find certain uses of copyrighted material to be noninfringing; its rationale being that when the free flow of information is sufficiently vital, it should override a copyright holder's interest in exclusive control. It represents an acknowledgement that prohibiting all copying whatsoever would stifle the free flow of ideas without serving any legitimate interest of the copyright holder. Under this doctrine, an individual in rightful possession of a copy of a work would then be permitted to undertake the necessary efforts to understand its ideas, processes, and methods of operation.
Buckman, supra note 2, at § 2(a).

149 Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 432 (1984).

150 Sega, 977 F.2d at 1523.

151 Id. (citing 17 U.S.C. § 107).

152 Sega, 977 F.2d at 1523.

153 Id.

[154] Id. at 1522. See also Connectix, 203 F.3d at 606.

[155] Connectix, 203 F.3d at 606; Sega, 977 F.2d at 1522.

[156] Sega, 977 F.2d at 1523.

[157] Id.

[158] Connectix, 203 F.3d at 607.

[159] Sega, 977 F.2d at 1523.

[160] Connectix, 203 F.3d at 606.

[161] Id.

[162] Id.

[163] Id.

[164] See id.

[165] See Connectix, 203 F.3d at 606; Sega, 977 F.2d at 1523.

[166] 17 U.S.C. § 107(2) (2000).

[167] 17 U.S.C. § 102(b) (2000).

[168] Sega, 977 F.2d at 1524 (quoting Baker v. Selden, 101 U.S. 99, 104 (1879)).

[169] Id.

[170] Id.

[171] Id.

[172] Id. at 1524-25 (quoting Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1236 (3d. Cir. 1986)).

[173] Whelan Assos., 797 F.2d at 1237.

174 Id.

175 See, e.g., Sega, 977 F.2d at 1525.

176 Id.

177 Id.

178 See Connectix, 203 F.3d at 603.

179 Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc, 307 F.3d 197, 206 (3d Cir. 2002).

180 Id. at 209-10.

181 Id. at 206 (citing 17 U.S.C. § 106).

182 Dun & Bradstreet, 307 F.3d at 211-12.

183 Id. at 214-15. Thus, with respect to any integral aspects of Company A's APIs that Company B may have copied during the reverse engineering process, Company A has a strong argument for copyright infringement based on Company B's creation of a derivative work. See id.

184 See Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1993).

185 See Dun & Bradstreet, 307 F.3d at 209-12.

186 Connectix, 203 F.3d at 605-06 (citing 17 U.S.C. § 107(3)).

187 See, e.g., id. at 606.

188 Id.

189 Id.

190 Sega, 977 F.2d at 1526. See also Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 449-50 (1984) (noting that copying of an entire work does not preclude fair use).

191 Universal City Studios, 464 U.S. at 449-50; Sega, 977 F.2d at 1526.

192 Dun & Bradstreet, 307 F.3d at 211-12.

193 Sega, 977 F.2d at 1523.

194 Id.

195 Id.

196 Id.

197 Id.

198 Id.

199 Sega, 977 F.2d at 1524.

200 Connectix, 203 F.3d at 607.

201 Id.

202 See Triad Sys. Corp. v. Se. Express Co., 64 F.3d 1330, 1336 (9th Cir. 1995).

203 Id.

204 Id. at 1336-37.

205 Id. at 1337.

206 Id. (quoting from Appellants's Excerpts of Record).

207 See id.

208 Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 843 (Fed. Cir. 1992).

209 Id. (citing Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 451 (1984)).

210 Atari, 975 F.2d at 843.

211 Id.

212 Id. In this respect, to the extent that Company B made intermediate copies of Company A's APIs for the purpose of identifying the functional requirements for compatibility, a fair use defense is likely to be upheld. See id. On the other hand, if Company B's use

of intermediate copies went beyond the bounds of this narrowly defined doctrine, Company A is likely to prevail on a copyright infringement claim. See id.

213 See supra Sections IV - V.

214 See supra Section IV.B.

215 See supra Section IV.B.

216 See supra Section IV.A.

217 See supra Section IV.A.

218 See supra Section V.

219 See supra Section V.