Software Derivative Work: A Circuit Dependent Determination

Dan Ravicher[*]

November 8, 2002

Free / Open Source Software ("FOSS") licensing relies critically on the concept of derivative work since software that is independent, i.e. not derivative, of FOSS need not abide by the terms of the applicable FOSS license. As much is required by § 106 of the Copyright Act[1] and admitted by FOSS licenses, such as the GNU General Public License, which states in Section 0 that "a 'work based on the Program' means either the Program or any derivative work under copyright law." It is being a derivative work of FOSS that triggers the necessity to comply with the terms of the FOSS license under which the original work is distributed. Therefore, one is left to ask, just what is a "derivative work?" The answer to that question depends on which court is being asked.

The analysis below sets forth the differing definitions of derivative work by Circuit. The broadest and most established definition of derivative work for software is the abstraction, filtration, and comparison test ("the AFC test") as created and developed by the Second Circuit. Some Circuits, including the Ninth Circuit and the First Circuit, have either adopted narrower versions of the AFC test or have expressly rejected the AFC test in favor of a narrower standard. Further, several other Circuits have yet to adopt any definition of derivative work for software.

As an introductory matter, it is important to note that literal copying of a significant portion of source code is not always sufficient to establish that a second work is a derivative work of an original program. Conversely, a second work can be a derivative work of an original program even though absolutely no copying of the literal source code of the original program has been made. This is the case because copyright protection does not always extend to all portions of a program's code, while, at the same time, it can extend beyond the literal code of a program to its non-literal aspects, such as its architecture, structure, sequence, organization, operational modules, and computer-user interface.

<u>The Copyright Act</u>

The copyright act is of little, if any, help in determining the definition of a derivative work of software. However, the applicable provisions do provide some, albeit quite cursory, guidance. Section 101 of the Copyright Act sets forth the following definitions:

> A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.
>
> A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical

---

[*] Associate, Patterson, Belknap, Webb and Tyler, LLP, New York, NY. The views expressed herein are personal to the author and should not be imputed to any of his clients or his firm.

[1] 17 U.S.C. § 106 (2002).

> arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a "derivative work".

These are the only provisions in the Copyright Act relevant to the determination of what constitutes a derivative work of a computer program. Another provision of the Copyright Act that is also relevant to the definition of derivative work is § 102(b), which reads as follows:

> In 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 work

Therefore, before a court can ask whether one program is a derivative work of another program, it must be careful not to extend copyright protection to any ideas, procedures, processes, systems, methods of operation, concepts, principles, or discoveries contained in the original program. It is the implementation of this requirement to "strip out" unprotectable elements that serves as the most frequent issue over which courts disagree.

<u>Abstraction, Filtration Comparison Test</u>

As mentioned above, the AFC test for determining whether a computer program is a derivative work of an earlier program was created by the Second Circuit[2] and has since been adopted in the Fifth[3], Tenth[4] and Eleventh[5] Circuits. Under the AFC test, a court first abstracts from the original program its constituent structural parts. Then, the court filters from those structural parts all unprotectable portions, including incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain. Finally, the court compares any and all remaining kernels of creative expression to the structure of the second program to determine whether the software programs at issue are substantially similar so as to warrant a finding that one is the derivative work of the other.

Often, the courts that apply the AFC test will perform a quick initial comparison between the entirety of the two programs at issue in order to help determine whether one is a derivative work of the other. Such an holistic comparison, although not a substitute for the full application of the AFC test, sometimes reveals a pattern of copying that is not otherwise obvious from the application of the AFC test when, as discussed below, only certain components of the

---

[2] *Computer Associates Intl., Inc. v. Altai, Inc.*, 982 F.2d 693 (2nd Cir. 1992).
[3] *Engineering Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335 (5th Cir. 1994); *Kepner-Tregoe, Inc. v. Leadership Software, Inc.*, 12 F.3d 527 (5th Cir. 1994).
[4] *Gates Rubber Co. v. Bando Chem. Indust., Ltd.*, 9 F.3d 823 (10th Cir. 1993); *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366 (10th Cir. 1997).
[5] *Bateman v. Mnemonics,Inc.*, 79 F.3d 1532 (11th Cir. 1996); *Mitek Holdings, Inc. v. Arce Engineering Co., Inc.*, 89 F.3d 1548 (11th Cir. 1996).

original program are compared to the second program. If such a pattern is revealed by the quick initial comparison, the court is more likely to conclude that the second work is indeed a derivative of the original.

Abstraction

The first step courts perform under the AFC test is separation of the work's ideas from its expression. In a process akin to reverse engineering, the courts dissect the original program to isolate each level of abstraction contained within it. Courts have stated that the abstractions step is particularly well suited for computer programs because it breaks down software in a way that mirrors the way it is typically created. However, the Courts have also indicated that this step of the AFC test requires substantial guidance from experts, because it is extremely fact and situation specific.

By way of example, one set of abstraction levels is, in descending order of generality, as follows: the main purpose, system architecture, abstract data types, algorithms and data structures, source code, and object code. As this set of abstraction levels shows, during the abstraction step of the AFC test, the literal elements of the computer program, namely the source and object code, are defined as particular levels of abstraction. Further, the source and object code elements of a program are not the only elements capable of forming the basis for a finding that a second work is a derivative of the program. In some cases, in order to avoid a length factual inquiry by the court, the owner of the copyright in the original work will submit its own list of what it believes to be the protected elements of the original program. In those situations, the court will forgo performing its own abstraction, and proceed to second step of the AFC test.

Filtration

The most difficult and controversial part of the AFC test is the second step, which entails the filtration of protectable expression contained in the original program from any unprotectable elements nestled therein. In determining which elements of a program are unprotectabe, courts employ a myriad of rules and procedures to sift from a program all the portions that are not eligible for copyright protection.

First, as set forth in § 102(b) of the Copyright Act, any and all ideas embodied in program are to be denied copyright protection. However, implementing this rule is not as easy as it first appears. The courts readily recognize the intrinsic difficulty in distinguishing between ideas and expression and that, given the varying nature of computer programs, doing so will be done on an ad hoc basis. The first step of the AFC test, the abstraction, exists precisely to assist in this endeavor by helping the court separate out all the individual elements of the program so that they can be independently analyzed for their expressive nature.

A second rule applied by the courts in performing the filtration step of the AFC test is the doctrine of merger, which denies copyright protection to expression necessarily incidental to the idea being expressed. The reasoning behind this doctrine is that when there is only one way to express an idea, the idea and the expression merge, meaning that the expression cannot receive copyright protection due to the bar on copyright protection extending to ideas. In applying this doctrine, a court will ask whether the program's use of particular code or structure is necessary for the efficient implementation of a certain function or process. If so, then that particular code or structure is not protected by copyright and, as a result, it is filtered away from the remaining protectable expression.

A third rule applied by the courts in performing the filtration step of the AFC test is the doctrine of scenes a faire, which denies copyright protection to elements of a computer program that are dictated by external factors. Such external factors can include: (a) the mechanical specifications of the computer on which a particular program is intended to operate; (b) compatibility requirements of other programs with which a program is designed to operate in conjunction; (c) computer manufacturers' design standards; (d) demands of the industry being serviced; and (e) widely accepted programming practices within the computer industry. Any code or structure of a program that was shaped predominantly in response to these factors is filtered out and not protected by copyright. Lastly, elements of a computer program are also to be filtered out if they were taken from the public domain or fail to have sufficient originality to merit copyright protection.

Portions of the source or object code of a computer program are rarely filtered out as unprotectable elements. However, some distinct parts of source and object code have been found unprotectable. For example, constants, the invariable integers comprising part of formulas used to perform calculations in a program, are unprotectable. Further, although common errors found in two programs can provide strong evidence of copying, they are not afforded any copyright protection over and above the protection given to the expression containing them.

Comparison

The third and final step of the AFC test entails a comparison of the original program's remaining protectable expression to a second program. The issue will be whether any of the protected expression is copied in the second program and, if so, what relative importance the copied portion has with respect to the original program overall. The ultimate inquiry is whether there is "substantial" similarity between the protected elements of the original program and the potentially derivative work. The courts admit that this process is primarily qualitative rather than quantitative and is performed on a case-by-case basis. In essence, the comparison is an ad hoc determination of whether the protectable elements of the original program that are contained in the second work are significant or important parts of the original program. If so, then the second work is a derivative work of the first. If, however, the amount of protectable elements copied in the second work are so small as to be *de minimis*, then the second work is not a derivative work of the original.

Analytic Dissection Test

The Ninth Circuit has adopted the analytic dissection test to determine whether one program is a derivative work of another.[6] The analytic dissention test first considers whether there are substantial similarities in both the ideas and expressions of the two works at issue. Once the similar features are identified, analytic dissection is used to determine whether any of those similar features are protected by copyright. This step is the same as the filtration step in

[6] *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435 (9th Cir. 1994).

the AFC test. After identifying the copyrightable similar features of the works, the court then decides whether those features are entitled to "broad" or "thin" protection. "Thin" protection is given to non-copyrightable facts or ideas that are combined in a way that affords copyright protection only from their alignment and presentation, while "broad" protection is given to copyrightable expression itself. Depending on the degree of protection afforded, the court then sets the appropriate standard for a subjective comparison of the works to determine whether, as a whole, they are sufficiently similar to support a finding that one is a derivative work of the other. "Thin" protection requires the second work be virtually identical in order to be held a derivative work of an original, while "broad" protection requires only a "substantial similarity."

No Protection for "Methods of Operation"

The First Circuit expressly rejected the AFC test and, instead, takes a much narrower view of the meaning of derivative work for software. The First Circuit holds that "method of operation," as used in § 102(b) of the Copyright Act, refers to the means by which users operate computers.[7] More specifically, the court held that a menu command hierarchy for a computer program was uncopyrightable because it did not merely explain and present the program's functional capabilities to the user, but also served as a method by which the program was operated and controlled. As a result, under the First Circuit's test, literal copying of a menu command hierarchy, or any other "method of operation," can not form the basis for a determination that one work is a derivative of another. It is also reasonable to expect that the First Circuit will read the unprotectable elements set forth in § 102(b) broadly, and, as such, promulgate a definition of derivative work that is much narrower than that which exists under the AFC test.

No Test Yet Adopted

Several circuits, including most notably the Fourth and Seventh, have yet to declare their definition of derivative work and whether or not the AFC, Analytic Dissection, or some other test best fits their interpretation of copyright law. Therefore, uncertainty exists with respect to determining the extent to which a software program is a derivative work of another in those circuits. However, one may presume that they would give deference to the AFC test since it is by far the majority rule amongst those circuits that have a standard for defining a software derivative work.

*   *   *

[7] *Lotus Development Corp. v. Borland Int'l., Inc.*, 49 F.3d 807 (1st Cir. 1995).