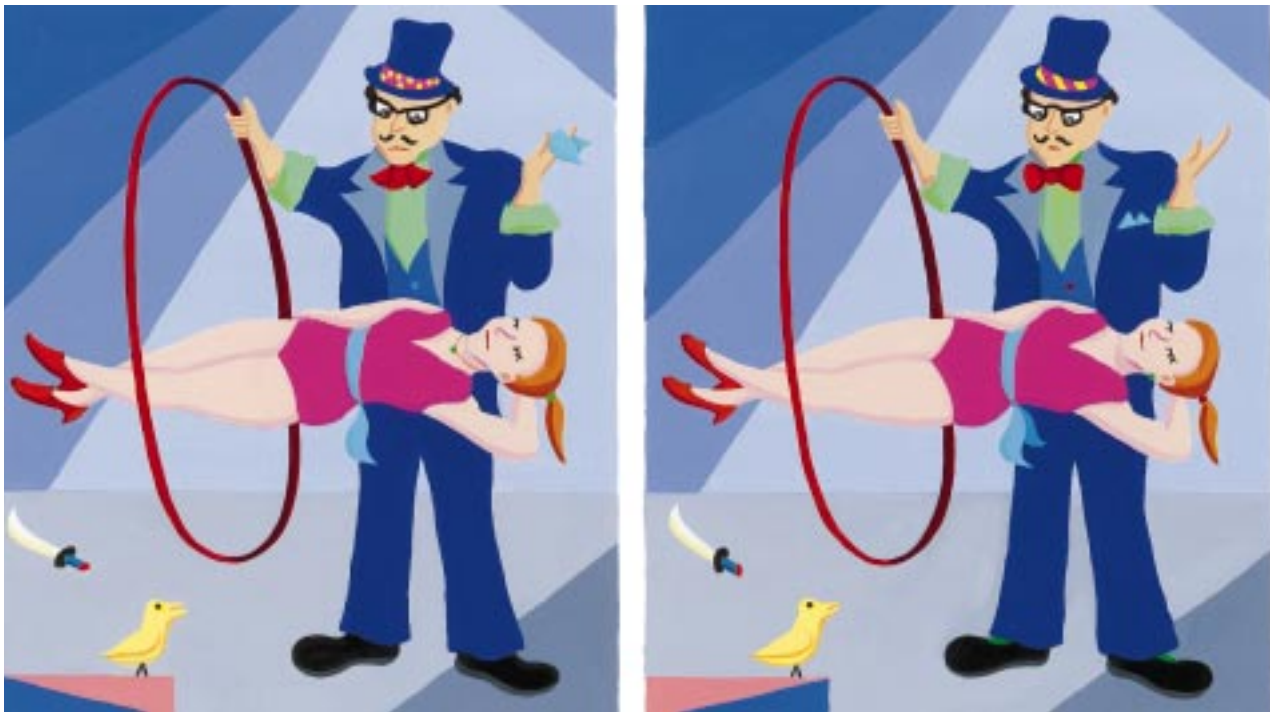

Whatever you do, make sure a derivative product is not too much like the original.

Practical Legal Aspects of SOFTWARE REVERSE ENGINEERING

Brian C. Behrens and Reuven R. Levary

A significant amount of litigation involving computer software focuses on reverse engineering. Unfortunately, despite this litigation, reverse engineering is probably one of the least understood concepts by the courts, legal counsel, and executives in the computer industry [3].

The U.S. Supreme Court has defined reverse engineering as "a fair and honest means of starting with the known product and working backwards to divine the process which aided in its development or manufacture." Additionally, a U.S. District Court in 1989 defined it as "the process of starting with a finished product and working backwards to analyze how the product operates or how it was made." Essentially, reverse engineering is the method by which programmers study an existing program in machine readable code by breaking it down into human readable form to create a similar product or one that can be used in conjunction with the existing software. This process of converting a program from object code back into source code is known as "decompilation," or disassembly.

Some software developers claim reverse engineering is unlawful, because it is far too easy to take an existing program, use reverse engineering to dis-

MASAKO EBATA

cover how it operates, make slight modifications to the original version, and market the amended version as a new product. Indeed, if after reverse engineering a program, a second program is developed that is substantially similar to the original with more in common than just its functional operations, it is most likely a case of software copyright infringement.

More recent cases, such as Sega Enterprises Ltd. v. Accolade, Inc. [10] and Atari Games Corp. v. Nintendo of America, Inc. [1], should allay this fear, as certain requirements must now be met before an altered program can be marketed as a "new" product. Not all reverse engineering efforts are illegal; most are not. Programmers use reverse engineering for numerous reasons; producing a competitive program is only one of them. For example, reverse engineering is used for breaking down software for the purposes of teaching students how to write code; for repairing malfunctioning software; to produce similar software to run on a different system; to modify a program for use on one's own computer; and to develop software that operates in conjunction with the original software [8].

To be on the safe side, be sure to avoid copying any program components that relate to a program's expression, or its aesthetic qualities.

The Fair Use Defense

Until some sort of legislative scheme is developed to deal with the discrepancies in our current system, how should attorneys advise their clients who are contemplating reverse engineering a software product? Although there is no certainty in this area, recent case law has made it prudent to recognize several caveats when undertaking a reverse engineering project. First, the attorney should advise the client to obtain an authorized copy of the software in question. For example, one hurdle preventing Atari from successfully asserting the fair use defense to counter its software infringement charge was the fact that it had "unclean hands" because it obtained a copy of Nintendo's source code by misrepresenting to the Copyright Office that it needed the code for litigation in which it was involved [2]. The federal circuit advised others that "[t]o invoke the fair use exception, an individual must possess an authorized copy of a literary work" [2, 6]. In order to avoid tainting a fair use defense, a software developer should obtain a software program it wants to reverse engineer from the open market, just as any other consumer would.

Additionally, one must be careful not to violate any licensing agreements signed directly with a computer company. For example, when a company, like Sega, licenses other software companies to use its code to produce games compatible with the Sega video game system, these companies often must promise not to attempt to reverse engineer any of Sega's codes for use in other games. Courts want to uphold such agreements and will probably view any reverse engineering in this circumstance as tainting a later fair use defense. Remember, the recent trend in court decisions is to accept the fair use defense and allow computer companies to reverse engineer software to advance technology and create market competition for software—but not at the expense of violating the more important public policy of avoiding fraud and misrepresentations while encouraging fair play.

Second, attorneys should advise their clients to be sure there is no means to obtain the information in the software other than by reverse engineering the program. The court in the Sega case specifically pointed out this caveat by stating, "[w]e conclude that where disassembly is the only way to gain access to the ideas and functional elements embodied in a copyrighted computer program . . . disassembly is a fair use of the copyrighted work, as a matter of law" [4]. Indeed, some companies publish certain code to allow for the development of compatible software. Therefore, if the source code for a particular function or process is obtainable by any other legal means, all these avenues should be explored before reverse engineering is undertaken.

Third, if reverse engineering can be used out of necessity, engineers should be advised to reverse engineer only the portions of the original program needed to decipher the precise functional elements required for the new program. Courts become wary of reverse engineering efforts that use portions of the original program to the extent the newly developed

software is similar in expression to the original. In Lotus Development Corp. v. Paperback Software International [5], the court held that Paperback infringed on Lotus's copyrighted software by devising its spreadsheet to have a look and feel similar to Lotus's popular 1-2-3 spreadsheet. Rather than copy only the functional elements of 1-2-3, Paperback had essentially copied much of Lotus's aesthetic qualities as well. While it is difficult to say when a court would consider two programs to be too much alike, the court in the Sega case attempted to give a few examples of where reverse engineering may have resulted in more than necessary portions of software being copied. Unfortunately, however, these examples are somewhat vague and not readily applied to everyday situations with confidence [7, 11]. Thus, to be on the safe side, be sure to avoid copying any program components that relate to a program's expression, or its aesthetic qualities.

Fourth, software developers should be sure to divide their reverse engineering efforts between two groups of engineers/programmers—one group to reverse engineer the program, the other to develop the new software. This method of ensuring "clean hands" is recommended so if a software developer is later charged with software copyright infringement, the company can produce records showing the newly developed program did not involve directly copying the original code. Specifically, this process begins with the first group of programmers reverse engineering the original software into the source code, so it can be read by human programmers. Next, the first group explains in a written journal or log the functions of the original program, as well as the ideas the program uses, without describing the expressive content of how the program will look to users. This journal is then given to the second group of programmers who attempt to design a program emulating the same functions and ideas. The second group cannot communicate directly with the original group, thus helping insulate the development process from any direct copying of the code in the original program. This process should ensure that the end product of the newly developed program looks somewhat different from the original program that was reverse engineered. The new program can then be marketed directly against the original program with little fear of violating the original's copyright protection [3, 9].

Fifth, despite following all these precautions, software developers should conduct research on the product to be reverse engineered to ensure that patent law does not provide protection for the particular process or function that is to be reverse engineered and used in a new program. Keep in mind that even though copyright law cannot protect functions and ideas, patent law does. Due to the rigorous requirements that must be met in order to get a patent on a certain function or process, few programs are patented. Nevertheless, a software developer can protect itself from patent infringement lawsuits by inquiring into this area before reverse engineering and marketing a new program.

Prudent software developers should consult with intellectual property attorneys to be sure they are in compliance with the latest case law. In this rapidly progressing area of law, new cases are being ruled on every day. As a result, the law could shift suddenly. C

References

1. Atari Games Corp. v. Nintendo of American, Inc., 975 F.2d 832 (Fed. Cir. 1992).
2. Atari, 975 F. 2d at 843. The court noted that [b]ecause Atari was not in authorized possession of the Copyright Office copy of 10NES, any copying or derivative copying of 10NES source code from the Copyright Office does not qualify as a fair use."
3. Davis, G., III. Scope of protection of computer-based works: Reverse engineering clean rooms and decompilation. In *15th Annual Computer Law Institute. Handbook Series (Patent, Copyright, Trademarks, and Literary Property Course)*, 1993, pages 1–15.
4. 977 F. 2d at 1527.
5. 740 F. Supp. 37 (D. Mass., 1990).
6. Harper & Row, 471 U.S. at 562-63 (where the court noted that knowing exploitation of purloined manuscript was not compatible with "good faith" and "fair dealings" underpinnings of fair use doctrine).
7. Hayes, D. The legality of disassembly of computer programs. *Comput./Law J. 14*, 1 (Jan. 1992), 4–12.
8. Lewis Galoob Toys, Inc. v. Nintendo of America, Inc., 780 F. Supp. 1283 (N.D. Cal. 1991), aff'd 964 F.2d 965 (9th Cir.1992) (where Galoob manufactured the Game Genie, which attached to Nintendo's video game cartridges and modified the game temporarily to alter certain game aspects, such as allowing the player to obtain more "lives" in a street battle. The court held reverse engineering to be an appropriate use in order to design this attachment to work with Nintendo's games).
9. McCabe, P. Reverse engineering of computer software: A trap for the unwary? *Comput. L. Assoc. Bul. 1*, 2 (Feb. 1994), 1–15.
10. Sega Enterprises Ltd. v. Accolade, Inc., 977 F. 2d 1510 (9th Cir. 1992).
11. Whelan Associates, Inc. v. Jaslow Dental Lab, Inc., 797 F. 2d 1222 (3rd Cir. 1986), cert denied, 479 U.S. 1031 (1987) (where the court devised an approach to differentiate protected expression from unprotected functions and ideas, an approach that has been roundly criticized).

Brian C. Behrens (bcb@suelthauswalsh.com) is an attorney in the law firm Suelthaus & Walsh, P.C., specializing in business law in St. Louis, Mo.

Reuven R. Levary (levarypr@sluvca.slu.edu) is a professor of decision sciences in the Department of Decision Sciences and MIS at Saint Louis University in St. Louis, Mo.