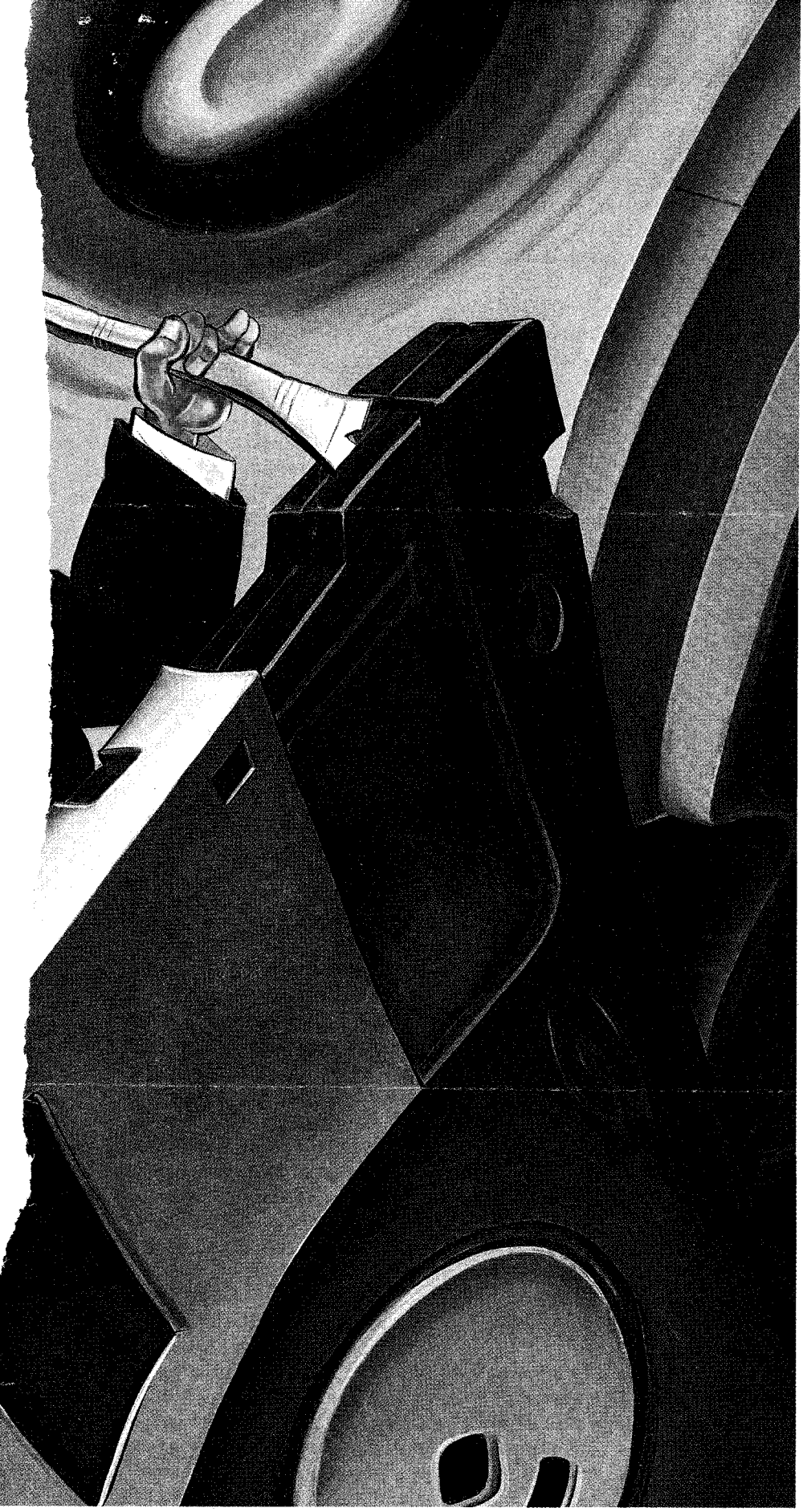

BY WILLIAM S. COATS,
HEATHER D. RAFTER

Instructions for disassembly

10 pitfalls for a client to sidestep when reverse engineering

As a kid, did you ever take an old watch apart to see what made it tick? Or disassemble a motor to see how the parts worked together to make the whole thing hum? You were doing an informal, at-home style of what industry calls reverse engineering.

Fast-forward to 1994.

You are a lawyer representing a company whose lifeblood is computer software. Its researchers are doing their own, supersophisticated brand of reverse engineering. They are dissecting a competitor's new and popular software program, trying to crack what is known as the object code to learn how the program works. This is a difficult and time-consuming job for any company, and one that often fails.

How far can the engineers go in unraveling the inner structure of

their competitor's product? At what point do they cross over the line from legitimate competition and fair use, to illegitimate copying or infringement of protected expression?

Fortunately, the answers to such dilemmas are becoming clearer. In a flurry of recent cases, courts have addressed the complex issues surrounding reverse engineering of computer software and hardware, outlining what companies can and cannot do. These decisions, emanating from several different courts, potentially shift billions of dollars of revenue among hardware and software makers.

Billions of dollars are at stake.

While the decisions are diverse, we can detect in them some guiding principles for business lawyers and their corporate clients. The underlying legal issues include:

- the question of whether there is any way other than reverse engineering to gain access to the functional elements of a program;
- the role of intermediate copying as a step in the whole process, even when the final product is not much like the competitor's copyrighted product;
- prohibitions in license agreements against reverse engineering;
- excessive copying of a competitor's program.

An industry grows up

Recent crucial cases concerning reverse engineering include *Sega Enterprises Ltd. v. Accolade Inc.*, 977 F.2d 1510 (9th Cir. 1992); *Atari Games Corp. v. Nintendo of America Inc.*, 975 F.2d 832 (Fed. Cir. 1992); and *Brooktree Corp. v. Advanced Micro Devices Inc.*, 977 F.2d 1555 (Fed. Cir. 1992).

These cases are a natural outgrowth of the fundamental changes to the copyright laws enacted in 1976 and 1980 and to the passage in 1984 of the Semiconductor Chip Protection Act. As a way to nurture the then-immature semiconductor and computer industries, Congress plainly was trying by these actions to foster entrepreneurial efforts and encourage what often amounted to risky investments.

Having matured for the most part, these industries now seek to protect their technological investments. An anemic overall economy has also prompted software and semiconductor companies to wring even greater revenue from their investments, to minimize uncompensated leakages of technological and manufacturing know-how, and to adopt more high-risk strategies to enhance profitability. Against this industrial and economic backdrop, the *Accolade*, *Brooktree* and *Atari Games* cases and their predecessors offer useful guidance to lawyers trying to help their clients successfully navigate the legal shoals of reverse engineering.

Critical differences

Before we review the court cases, it will help to get a layperson's-eye-view of the mechanics of reverse engineering. We need to keep in mind a couple critical distinctions about this process.

First, the information sought by reverse engineers varies greatly according to whether software (a program) or hardware (a chip) is the object of their efforts.

Second, a pirate and a legitimate reverse engineer are an ocean apart. A pirate is someone merely intending to copy, in its entirety, someone else's program or chip in order to usurp market share. A pirate is a thief — no more, no less. On the other hand, a legitimate reverse engineer is trying to develop compatible or competitive products. The cases we will discuss are generally about legitimate reverse engineers who ran afoul of the law despite their good-faith attempts at competition.

How it's done

Reverse engineering of software

To understand software reverse engineering, we first need to look at what programmers do.

Programmers write software programs in high-level "source code" languages using alphanumeric characters understandable to humans. A programmer routinely includes labels and comments in a program to identify parts and to explain what the program is doing.

This source code must then be transformed into object code, which a computer can understand. It is a process known as "assembly" or "compilation," and it results in a significantly different version of the program. Assembly involves more than just converting the alphanumeric source code into ones and zeroes. The source code's labels and comments, which perform no useful function for the computer, are eliminated. Additional program instructions that are functionally required for operation with a particular type of microprocessor may be added to the program automatically by the compiler. Further, the program instructions may be "optimized" — meaning they are placed in an order more usable by the computer. The *published* object code version thus is significantly different from the source code version.

To reverse this process, the first step in "reading" the object code version is to disassemble or decompile it. A program written in object code is one long string of "bits" — zeroes and ones. The bits are organized into "bytes," each byte being a unit of computer memory composed of eight "bits" — eight ones

Coats is a partner in the Palo Alto office of Brown & Bain. Rafter is an associate in the San Francisco office of Gibson, Dunn & Crutcher. They represented Accolade in Sega v. Accolade.

and/or zeroes. A byte contains the binary representation of a number from 0 through 255. This byte can represent numerical, graphical or textual information, or a computer instruction.

Just by looking at this string of zeroes and ones, a human reader has no way of knowing what they represent. A reverse engineer therefore must translate the ones and zeroes into assembly language, a low-level programming language that humans can understand.

Such a project would be impossible to accomplish solely from human memory without making copies. Thus, computers are invariably used for disassembly. The object code is copied into a computer's memory, and the machine translates it into human-readable assembly language.

A human programmer must still study the assembly language extensively and interpret it to understand how the program operates. The end product is not the original source code, but a reverse engineer's interpretation of how the program operates.

There are various ways to obtain the object code. In the *Accolade* case, Accolade got it by attaching to the computer chip a commercially available device called an emulator, which "read" the object code and downloaded it into a computer. In the *Atari Games* and *Brooktree* cases, the reverse engineers "stripped" or "peeled" the chip to read the object code. In *Atari Games* and *Accolade*, the object code was then disassembled through a commercially available program that translates the object code into assembly language. Human programmers then read the assembly language to understand the object code.

Accolade set the parameters for the circumstances under which reverse engineering would qualify as a fair use. The Ninth Circuit opinion said: "We conclude that where disassembly is the only way to gain access to the ideas and functional elements embodied in a copyrighted computer program and where there is a legitimate reason for seeking such access, disassembly is a fair use of the copyrighted work, as a matter of law."

The underlying principle is that the copyright laws should not be used to discourage legitimate competition.

Reverse engineering of chips

The Semiconductor Chip Protection Act (SCPA) is a comprehensive law specifically intended to protect the design layout of chips, which typically consist of multiple layers of electrical connections built up by photographic techniques. Neither the Copyright Act nor the Patent Act was considered adequate to protect proprietary rights in chips.

The SCPA prohibits the copying of original chips but excludes from protection chips that are not original or consist merely of designs that are "staple, commonplace or familiar in the semiconductor industry." The SCPA specifically permits reverse engineering for the "purpose of teaching, analyzing or evaluating the concepts or techniques embodied" in the chip or the "circuitry, logic flow or organization of components used."

In *Brooktree*, the purpose of the reverse engineering by Advanced Micro Devices was to make a semiconductor chip functionally equivalent to Brooktree's. To create a competitive product, Advanced Micro Devices needed to understand the code in the chip necessary for its operation.

The Top 10 Pitfalls

The *Accolade* case tells us what reverse engineers are allowed to do, but *Brooktree*, *Atari Games* and other cases provide a list of legal "don'ts." We now offer, in David Letterman style, our list of the Top 10 "Don'ts."

10. Don't copy the I Ching (*Walker v. University Books*).

By its very nature, reverse engineering of software requires making copies on the way to the final product. The threshold question in any discussion of the legality of reverse engineering of software is whether making such intermediate copies is itself an infringement.

In *Walker v. University Books*, 602 F.2d 859 (9th Cir. 1979), plaintiff, a designer of I-Ching fortune-telling cards, brought a copyright infringement action against defendant, who had prepared camera-ready mechanicals for its own set of I-Ching cards that were allegedly copies of plaintiff's cards. The defendant had not yet printed any of the final, allegedly infringing cards, but had merely created the mechanical drawing that would be photographed to make the plates, from which it could ultimately print the cards. Nevertheless, the complaint alleged that the final product that would flow from the mechanical drawing would be a duplicate of plaintiff's copyrighted cards.

The Ninth Circuit held in *Walker* that copyright infringement could be based on the camera-ready mechanicals for defendant's I-Ching cards, even though the actual cards had not yet been printed. In sum, the panel authorized plaintiffs to strike preemptively before defendants could complete their infringing conduct. The Ninth Circuit, however, remanded the case to the district court on the issue of substantial similarity of the final products.

Intermediate copies can cause a problem.

In the *Accolade* case, the Ninth Circuit reiterated that *Walker* set forth the general rule that intermediate copying can itself be the basis

for a copyright infringement action.

Thus, in the specific context of reverse engineering, intermediate copying alone can be actionable, regardless of whether the end product is substantially similar to the copyrighted work. The crucial distinction between *Walker* and *Accolade*, however, is that Accolade's intermediate copying was held to be a fair use because Accolade's ultimate purpose was lawful.

9. Don't copy "Pinocchio" (*Walt Disney Co. v. Filmation*).

A Central District of California case, *Walt Disney Productions v. Filmation*, also suggests that intermediate copying is actionable. *Filmation* involved the issue of whether defendant's production materials for its film, "The New Adventures of Pinocchio," might infringe Disney's copyrights in the film "Pinocchio." The underlying story of Pinocchio was in the public domain and, thus, not protectible. However, Disney's treatment of the story and related music, dialogue characterizations and art work were protectible.

Filmation's production materials included its script, storyboard, story reel and promotional trailer. The film, however, had not yet been completed and there was *no final product* that could be compared to Disney's film to make a determination of substantial similarity. The court held, following *Walker*, that these production materials could form the basis for a copyright infringement action regardless of whether they were ultimately used in an infringing motion picture.

Thus, *Filmation* teaches that infringement can be based not only on camera-ready mechanicals, which evidenced the final product, but also on preparatory materials that were not necessarily part of the final product. The *Filmation* decision, coupled with *Walker*, suggests that a reverse engineer may not avoid infringement simply by making enough changes to the copyrighted material of another so that the end product is not substantially similar. If a dispute arises, infringement could be found on the basis of intermediate copies if they are not protected by fair use or some other doctrine.

8. Don't sign an agreement prohibiting reverse engineering (*SAS*).

In *SAS Institute v. S&H Computer*, 605 F. Supp. 816 (1983) and its companion case, *S&H Computer v. SAS Institute*, 568 F. Supp. 416 (1983), the Middle District of Tennessee found that disassembly was actionable where defendant used copies of source code and object code obtained as a licensee of plaintiff.

If you are a licensee of a competitor whose program you intend to disassemble, you first need to examine the terms of the license agreement. If the agreement prohibits reverse engineering, don't do it. Even if it doesn't bar reverse engineering, you need to take the vital self-protective measure of starting and maintaining a clear paper trail and, through a "clean room" technique, isolating those employees who are familiar with your competitor's program, so that there can be no possibility of accidentally copying part of the examined work into your product.

One open question is whether a company could enforce a shrink-wrap license agreement prohibiting reverse engineering. So-called shrink-wrap licenses are included inside the packaging of a software product. The enforceability of such licenses has been the subject of much debate and scholarly commentary. The Third Circuit's decision in *Step-Saver Data Systems v. Wyse Technology*, 939 F.2d 91 (3d Cir. 1991), suggests shrink-wrap licenses might be unenforceable.

7. Don't hire reverse engineers with poor skills (*Brooktree v. AMD; Atari Games v. Nintendo*).

As we've mentioned, reverse engineering is a complex and often frustrating process for even the most proficient computer engineers. In *Sega v. Accolade*, for example, Accolade's engineers spent an estimated two man-years reverse-engineering Sega's console and game programs.

The pitfalls of employing unskillful reverse engineers is well illustrated by the *Atari Games* and *Brooktree* decisions, recently decided by the federal circuit. In *Atari Games*, the engineers stalled in their effort to reverse-engineer Nintendo's 10NES system and had to resort to means disapproved by both the district court and the federal circuit. *Atari Games*, 975 F.2d at 836, 843-44.

Astute computer programmers will make sure that they do not use protectible expression in the ultimate product. Hoisting this shield, they ward off the danger of copying. In *Brooktree*, the reverse engineers spent two years and more than $3 million trying to replicate the functions of the Brooktree chip. Ultimately, Advanced Micro Devices copied a large part of that